NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INFORMATION-SHARING APPLICATION STANDARDS FOR INTEGRATED GOVERNMENT SYSTEMS

by

Gary Lavers

December 2010

Thesis Advisor:	Robert Looney
Second Reader:	Sean Everton

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2010	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Information-Sharing Application Standards for Integrated Government Systems	5. FUNDING NUMBERS
6. AUTHOR(S) Gary C. Lavers	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

This thesis examines the Department of Homeland Security's (DHS) Homeland Security Information Network (HSIN) early acquisition shortcomings identified by the Government Accountability Office, DHS Inspector General and Congressional Research Service reports. Challenges identified in the initial development of HSIN reveal a lack of adequate program management, requirements planning, risk analysis and architectural design led to low user acceptance and continued DHS information-sharing challenges. Lessons learned from HSIN are examined to determine which best practices can help ensure major government software-acquisition projects meet user's needs. Often overlooked, but critical, software program-management practices include user requirements planning that focuses development on the highest priority tasks and encourages the timely accomplishment of project milestones, risk planning that ensures potential roadblocks are understood and addressed, and architectural design practices that foster the integration of both newly developed and legacy information systems. Without initial and continuous life-cycle requirements, risk and architectural planning, software projects run an increased risk of going over budget, missing operational milestones and ultimately not meeting its user's needs.

14. SUBJECT TERMS Homeland Security Information Network, HSIN, Software Acquisition, Homeland Defense, Enterprise Architecture, Software Reuse, Component Architecture, Interoperability, Extensibility, Service-Oriented Architecture, Software Risk Management, Requirements Planning	15. NUMBER OF PAGES 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

INFORMATION-SHARING APPLICATION STANDARDS FOR INTEGRATED GOVERNMENT SYSTEMS

Gary C. Lavers
Major, United States Air Force
B.S., University of Maryland, 1995

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF ARTS IN SECURITY STUDIES
(HOMELAND SECURITY AND DEFENSE)**

from the

**NAVAL POSTGRADUATE SCHOOL
December 2010**

Author: Gary C. Lavers

Approved by: Robert E. Looney, PhD
Thesis Advisor

Sean F. Everton, PhD
Second Reader

Harold A. Trinkunas, PhD
Chairman, Department of National Security Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis examines the Department of Homeland Security's (DHS) Homeland Security Information Network (HSIN) early acquisition shortcomings identified by the Government Accountability Office, DHS Inspector General and Congressional Research Service reports. Challenges identified in the initial development of HSIN reveal a lack of adequate program management, requirements planning, risk analysis and architectural design led to low user acceptance and continued DHS information-sharing challenges. Lessons learned from HSIN are examined to determine which best practices can help ensure major government software-acquisition projects meet user's needs. Often overlooked, but critical, software program-management practices include user requirements planning that focuses development on the highest priority tasks and encourages the timely accomplishment of project milestones, risk planning that ensures potential roadblocks are understood and addressed, and architectural design practices that foster the integration of both newly developed and legacy information systems. Without initial and continuous life-cycle requirements, risk and architectural planning, software projects run an increased risk of going over budget, missing operational milestones and ultimately not meeting its user's needs.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application Programmer Interface
ATIX	Anti-Terrorism Information Exchange
CBP	Customs and Border Protection
CCB	Change Control Board
CMMI	Capability Maturity Model Integration
CMS	Content Management System
CMS	Content Management System
COI	Communities of Interest
COP	Common Operating Picture
COTS	Commercial Off-the-Shelf Software
CRS	Congressional Research Service
DHS	Department of Homeland Security
DoD	Department of Defense
DOJ	Department of Justice
EA	Enterprise Architecture
EAF	Enterprise Architecture Framework
FAC	Federal Advisory Committee
FEA	Federal Enterprise Architecture
GAO	Government Accountability Office
HSIN	Homeland Security Information Network
HSINAC	Homeland Security Information Network Advisory Council
IEEE	Institute of Electronic Engineers
IG	Inspector General
ISCC	Information Sharing Coordination Committee
ISE	Information Sharing Environment
ISGB	Information Sharing Governance Board
IT	Information Technology
MOC	Mission Operators Committee
OMB	Office of Management and Budget
PM	Program Manager

RE	Requirements Elicitation
RFI	Request for Information
RISS	Regional Information Sharing System
SEI	Software Engineering Institute
SOA	Service-Oriented Architecture
SPMN	Software Program Manager Network
UI	User Interface
USAF	United States Air Force
XML	EXtensible Markup Language

I. INTRODUCTION

> One clear lesson of September 11 was the need to improve the sharing of information. To prevent further attacks and to protect the homeland, we need to stay a step ahead of those individuals and organizations intent upon harming America. Key to preventing future attacks is the gathering of information about terrorist risks and threats and then ensuring that the information gets into the hands of those whose responsibility it is to protect our communities and critical infrastructure.[1]

A. MAJOR RESEARCH QUESTION

One mandate of the Homeland Security Act of 2002 directed the Department of Homeland Security (DHS) to coordinate data and information sharing among federal agencies, local and state governments, and the private sector in order to counter terrorist threats and strengthen homeland security.[2] Driving this requirement was the fact that a large amount of government intelligence data is never processed, and a vast amount of potentially critical information "has undergone little or no assessment regarding its accuracy or implications."[3] Improving communications and information sharing between federal, state, and local agencies is a critical requirement established by the Homeland Security Act, and its implementation is part of the DHS's primary mission.

To meet these requirements, DHS deployed the Homeland Security Information Network (HSIN), with development beginning in 2002 and fielded in July 2004. This web-based portal application provides users with access to raw

---

[1] U.S. Executive Office of the President, "National Strategy for Information Sharing: Successes and Challenges in Improving Terrorism-Related Information Sharing," October 2007, 7, http://handle.dtic.mil/100.2/ADA473664.

[2] Homeland Security Information Sharing Act of 2002 (Public Law 107-296, November 25, 2002), 11.

[3] Harold Relyea and Jeffrey W. Seifert. Congressional Research Service, *Information Sharing for Homeland Security a Brief Overview.* Congressional Information Service, Library of Congress, 2005, 2.

collected data, subject analysis, document libraries, “chat” tools, and emergency management collaboration modules. However, due to shortcomings in the current system, the DHS has proposed a follow-on system dubbed “HSIN Next Generation,” and has subsequently stopped development of the first-generation system.[4]

At the request of Congress in 2007, members of the Government Accountability Office (GAO) presented the results of a comprehensive study to the House and Senate Homeland Security Committees. The information indicated that the original HSIN was poorly coordinated in its development, did not interface well with existing state and local data and applications, and contained unnecessary redundancy with programs used by regional centers.[5] The GAO study also claims that many of the flaws present in the development of HSIN, which hampered its widespread use, were also present in the development of the next-generation system currently under development.

This thesis examines the development and implementation of government information-sharing systems and looks specifically at the flawed acquisition and management processes that hamper information sharing, as well as best practices that facilitate collection, processing, and availability of government information. Proper acquisition and management of government systems requires careful assessment of existing technology, used by current and perspective users, along with a review of successful government and industry acquisition strategies.

This thesis specifically identifies shortcomings evident in the development and management of HSIN to order to answer the following questions:

1. What flaws existed in the development and deployment of the Homeland Security Information Network?

[4] The term *system* in this thesis refers to software and its related architecture and components and not to the hardware or network used to run the software.

[5] Relyea and Seifert. *Information Sharing for Homeland Security*, 3.

2. How can lessons learned from early management, planning and implementation of HSIN apply to the development of current and future systems?

3. What planning and management practices reduce cost and development time when considering user's needs and project risks during a program's life cycle?

4. What specific system design standards (component-based architectures, legacy system integration, service-oriented architecture, and other technologies) facilitate a cost-effective layered approach to developing flexible systems with a high likelihood of compatibility with future technology?

B. PROBLEMS AND HYPOTHESIS

In the past several years, numerous systems have been developed and implemented across the government in an effort to better share information. The question to ask now is whether interagency information sharing has adequately improved since 9/11. The answer seems to be weighted toward the negative. Amy Zegart writes in her book, *Spying Blind,* "Information sharing and analysis, two critical shortcomings raised in the wake of 9/11, have not improved much and in some cases have gotten worse."[6] She goes on to explain "…information is stored on nearly thirty separate, incompatible information networks. To access them all, analysts must use more than a half-dozen different computers stacked underneath their desks."[7] Zegart is not alone in her analysis. The Markle Foundation Task Force, one of the leading private advocates for intelligence

[6] Amy B. Zegart, *Spying Blind: The CIA, the FBI, and the Origins of 911* (Princeton, N.J.: Princeton University Press, 2007), 186.

[7] Ibid., 187.

information sharing, states that “today we are still vulnerable to attack because—as on 9/11—we are still not able to connect the dots.”[8]

The development of government systems requires careful assessment of existing technology for current and future users, and the implementation of component reuse and interoperability standards that facilitate simplified development and maintenance of the three application tiers (data, business logic, and user interface), as well as the ability to share data and functionality with external applications.

This thesis argues that newly developed government data-sharing systems require dedicated program management to ensure adequate initial and continuous life-cycle planning are an integral part of the system’s design. When time is critical, planning for user requirements, risk and architecture are sometimes neglected. This thesis shows that many of HSIN’s flaws are rooted in poor management practices that overlooked these critical aspects of systems design.

The following is a list of recommendations for the development and acquisition of new systems based on lessons learned from HSIN and other government systems, reports and studies conducted to address software acquisition shortcomings, as well as commercial and academic sources listed in the next chapter. At a minimum, any large user-based, government software acquisition strategy should have the following elements:

1. Assignment of a fulltime project manager and staff to oversee the acquisition and development process.
2. Use of best practices in requirements planning that ensures user and stakeholder needs are prioritized, modeled, validated and adequately resourced.[9]

[8] Markle Foundation Task Force, “Nation at risk policy makers need better information to protect the country,” 2009, 1, http://www.markle.org/events/20090310_nar/20090304_mtf_report.pdf.

3. Use of best practices in risk planning that covers the project's entire life cycle from initial planning to retirement in order to anticipate and mitigate potential problems.

4. Develop an architecture plan that considers interoperability, component reuse, extensibility and service-oriented design practices that help new systems integrate legacy data and increase the probability of compatibility with future technology.

C. METHODS AND SOURCES

This thesis utilizes a qualitative research approach to determine ways the government software acquisition process can be improved. This method provides a systematic approach to understand and evaluate complex and continually evolving information systems subject matter. The intent of this research is to determine where weaknesses exist in the government system acquisition process and, through a comprehensive review of current practices, offer methods to improve the process.

D. OVERVIEW

Following a literature review, the third chapter of this thesis consists of a case study that examines the planning, management and acquisition of HSIN and the mistakes made in the process. GAO, CRS and IG reports along with private foundation studies are considered to determine which best acquisition practices could have improved the development of HSIN.

Chapter IV begins the process of examining best practices that may have alleviated many of the program management challenges identified earlier in this thesis. The first of these is system acquisition requirements planning that help developers understand and document the core problem set that allows program managers to define minimum essential tasks and steer resources to where they

[9] The term *stakeholder* is used in this thesis to describe the organization accepting the delivered product, whether developed internally or contracted to a vendor.

are needed most. Practices like user elicitation, analysis and modeling, and continuous project requirements validation help keep the project's main tasks front and center. The chapter concludes with a case study of Intellipedia based on a paper by the CIA's Chief Technology Officer for the Center for Mission Innovation, Calvin Andrus. The development of Intellipedia shows how intelligence information sharing needs can be met by understanding the user needs and selecting the best software and system to match clearly defined requirements.

Chapter V addresses the final two challenges faced by HSIN, identified by GAO and IG and described in Chapter II; that is, risk planning and architectural design practices. Risk planning involves identifying potential problems that could occur during system development, determining the probability of a risk occurring and devoting time to create a plan to mitigate and respond to identified risks. Without risk planning, project managers and developers are forced to continuously fight fires instead of simply implementing alternative plans when contingencies occur. Finally, Chapter V closes with a discussion of modern componentized architectural design practices that ensure software is reusable, extensible, interoperable with external systems and able to communicate with legacy systems for data and functionality sharing.

II. LITERATURE REVIEW

Several U.S. government agencies, non-profit organizations, academic institutions, and standards organizations publish reports concerning the development of government information systems and software acquisition strategies. There are also several web-portals devoted to software acquisition best practices, past experience, and resources for integrating rapidly changing technology into developing and legacy systems. The best method available to build a comprehensive review of best practices is to extract information from these sources.

A. GOVERNMENT SOURCES

The most comprehensive work in assessing government acquisition of information-sharing systems comes from GAO and Inspector General (IG) reports. The bibliography section of this thesis provides several sources of GAO and IG reports detailing the shortcomings of HSIN's initial planning and development. These reports demonstrate that the main priority in creating a new system should be to build a prioritized functionality set based on, for example, an assessment of systems already in use (if any). Once developers and stakeholders agree to and document requirements, the choice to create new functionality or integrate existing technology can be evaluated. This process reduces the likelihood that duplicate systems are developed and that the end user's needs are ultimately satisfied.

The GAO has also provided some useful guidance concerning data collection and management, including XML metadata tagging technology that fosters information sharing between agencies.[10] In a report focusing on the

[10] XML is a plain-text data structure. Metadata is a method to include contextual information with raw data in the form of XML tags. Combined sources of metadata can be searched to provide data links and potentially produce information. Metadata can be transmitted and received between non-compatible systems because the tags are formed using XML that is readable across nearly all platforms.

integration of data technologies, the GAO recommends the development of information-sharing standards through web services and secure XML communication-based protocols that enable information sharing between existing systems. As an example, the report uses transportation department data combined with elevation, weather models, census, and infrastructure data from four agencies to analyze various response strategies to natural and manmade disasters. In order to do this, data must be tagged and made available through modern data agnostic techniques.[11]

The leading resource providers for best or "gold" software system management is Defense Department's Information Analysis Center[12] and Carnegie Melon's Software Engineering Institute (SEI), as well as several Department of Defense (DoD) armed services specific sites.[13] These organizations are dedicated to providing government and industry a single source repository of accepted and best practices for software system architecture, acquisition and management. Other sources of government system data integration include the Lessons Learned Information Sharing portal (LLIS.gov), the National Strategy on Information Sharing, and several Congressional Research Service (CRS) reports (these and others are listed in the bibliography section).

B. NON-PROFIT INSTITUTIONS

Respected non-profit groups such as the RAND Corporation and the Markle Foundation have provided valuable information-sharing and collaboration recommendations—some of which have been adopted by the federal government. Similar to the GAO and IG documents, these foundations have

[11] Randall A. Yim, United States General Accounting Office, "National Preparedness Integrating New and Existing Technology and Information Sharing Into an Effective Homeland Security Strategy," 2002, 8, http://purl.access.gpo.gov/GPO/LPS34938.

[12] The Data and Analysis Center for Software (n.d.), https://www.thedacs.com/.

[13] Software Engineering Institute, Carnegie Mellon (n.d.), http://www.sei.cmu.edu/.

produced reports critical of current government information-sharing systems but have also provided useful roadmaps for improving systems and managing future acquisitions.

One particularly useful set of reports comes from the Markle Foundation Task Force on National Security in the Information Age. To date, this privately funded task force has produced five reports emphasizing the need for better government information-sharing standards. In the foundation's 2006 report, "Mobilizing Information to Prevent Terrorism," a comprehensive technology review is included, laying a foundation for data interoperability: "One of the principal goals of networked information is to separate content from applications—i.e., to make information usable and interoperable across many applications and systems."[14]

The Markle Foundation also recommends against centralizing information, but rather the implementation of distributed component architecture that covers "different domains, each having different security and access requirements."[15] RAND and the Markle Foundation also cover constitutional law and public and private individual rights issues that should be considered when governments integrate data.

Another prominent organization is the Software Program Manager's Network (SPMN), dedicated to fixing what is broken with the government software acquisition process "when essential software disciplines and practices are not implemented on large-scale projects, complexity snowballs into chaos and cripples or kills programs."[16] One contribution to software acquisition strategies is the group's "16 Critical Software Practices," which specifically addresses underlying cost and schedule drivers that have caused many software

[14] John and Mary R. Markle Foundation, "Mobilizing Information to Prevent Terrorism: Accelerating Development of a Trusted Information Sharing Environment." The Markle Foundation, 2006, 58.

[15] Ibid.

[16] Software Program Manager's Network, "The Little Book of Bad Excuses," 1998, http://www.spmn.com/products_guidebooks.html.

intensive systems to be delivered "over budget, behind schedule and with significant performance shortfalls"—information extremely useful for this thesis.[17]

C. ACADEMIC AND STANDARDS BASED INSTITUTIONS

The purpose of government watchdog and accountability focused agencies like the GAO and IG is to review existing government programs and provide recommendations for improvement. Another approach is to study organizations that provide guidance for both existing and future system development, and recommend development strategies for user interface, business logic and data layers across the entire enterprise.

1. Enterprise and Service-Oriented Architectures

For information-sharing systems, one of the best resources is Carnegie Melon's Information Sharing Environment (ISE) Portal dedicated to both government and industrial software development and acquisition.[18] For example, the ISE's Enterprise Architecture Framework (EAF) and the Mission-Oriented Success Analysis and Improvement Criteria (MOSAIC) documents provide an excellent foundation for managing and planning system architecture that incorporates the latest requirements, risk and component-based design principles. One of many beneficial aspects of ISE's reports concerns leveraging existing capabilities across agencies that help find ways to collaborate using a broad systems approach to integrating legacy data. To do this, ISE recommends a service-oriented architecture (SOA) that allows agencies to expose their data and processes for use by other trusted systems. This method uses an agnostic data layer that eliminates compatibility issues across systems.[19]

---

[17] Software Program Manager's Network, "The Little Book of Bad Excuses," 1998, 1.

[18] Enterprise Architecture Framework Version 2.0, Information Sharing Environment, 2008, 1, http://www.ise.gov/docs/eaf/ISE-EAF_v2.0_20081021.pdf.

[19] Ibid., 4.

2. User Interface Guidelines

While there are seemingly countless academic articles and commercial studies related to system architecture and data, government and non-profit institutions listed so far generally lack recommendations for user interface design. Academia and commercially published books are a good source to bridge this gap.[20] Collaborative information systems used in industry and government include portals, wikis, content management systems (CMS), and mashups (to name a few). When employed effectively, well-designed user interface systems facilitate information sharing and collaboration, increase productivity, and aid the government in its need to facilitate end user communication. A poorly designed user interface can impede the flow of information and lower productivity. Making information searchable and combining data into useful information require solid technological backend design as well as an interface that makes utilizing the delivered information simple for the end user.

3. Complying with Standards

The software acquisition and project management process has been exhaustively studied and documented by industry, standards organizations and government for decades. The baseline standard for acquisition is the responsibility of the Institute of Electrical and Electronic Engineers (IEEE), and their work is regarded by both government and industry as the definitive source for standardizing many software and system design practices. For example, the IEEE document Standard 1062 provides a complete list of steps necessary to manage new software projects.[21]

---

[20] Several computer science and systems journals, including IEEE Computing Society, the SOA Magazine and the World Academy of Science, Engineering and Technology provide articles dedicated to user interface integration. Government sources primarily focus on data and business logic layers of software engineering.

[21] IEEE Computer Society. "Software Engineering Standards Committee. and Institute of Electrical and Electronics Engineers," IEEE Recommended Practice for Software Acquisition. (1994).

The IEEE also provides a single source of both academic and commercial thought on issues related to user interface. Several IEEE documents standardize the concept of application development using a reusable and layered approach to system design and interoperability. IEEE also recommends a model "designed in three layers: presentation, application (also called the business-logic layer), and data."[22] These three layers are combined using a component model of reusable parts that can be "plugged in" to other systems as required. For example, a security component can be built into the business logic layer that can be used across multiple applications. Improvements and bug fixes to the security components can then be applied across the enterprise without redesigning each individual system codebase.

At the user interface layer, component design gives users a common set of controls that require less training. Controls are easily integrated into the business logic and data layers of multiple systems via a common Application Programmer Interface (API) architecture. These "components are essentially characterized by an API…effectively standardizing UI integration."[23]

When combined, the standards organizations and best practice sources mentioned in this chapter form a solid foundation for sound system design, integration and acquisition strategies. Practices mentioned by these sources are used in Chapters IV and V to offer strategies that could have improved HSIN's initial acquisition, development and integration with legacy systems, and are useful for the acquisition of future systems.

[22] Marino Linaje, Juan Carlos Preciado, and Fernando Sanchez-Figueroa, "Engineering the Web Track – Engineering Rich Internet Application User Interfaces over Legacy Web Models," *IEEE internet computing* 11, no. 6 (2007): 53–59.

[23] Ibid.

III. HOMELAND SECURITY INFORMATION NETWORK

> We learned of the pervasive problems of managing and sharing information across a large and unwieldy government that had been built in a different era to confront different dangers.
>
> — 9/11 Commission Report

A harsh spotlight fell on the intelligence and law enforcement communities following the terrorist attacks on the United States on September 11, 2001. Each link in the chain that could have exposed or thwarted the attacker's plan has since been thoroughly dissected, analyzed and critiqued by intelligence experts around the world. A lack of adequate information sharing and collaboration between the various branches of the U.S. government is often cited as one of the major failures contributing to the success of the terrorist's operation that day.

To remedy these shortfalls, the U.S. government has taken broad steps to reform the intelligence community in order to improve information sharing. Since 9/11, an expansive intelligence legal framework has been implemented, including the passage of the Patriot Act in 2001, establishment of the Department of Homeland Security, creation of the Director of National Intelligence position and the creation of the National Counter Terrorism Center—all with the intent to improve intelligence gathering, sharing, analysis, and dissemination.

The requirement for networked systems to meet the information-sharing and collaboration needs of these newly created government structures spawned countless software and system acquisitions that attempt to combine legacy data into actionable information accessible by multiple agencies.[24] The Department of Homeland Security responded to the need to coordinate homeland security information by commissioning the web-based application, dubbed the Homeland

[24] The term "legacy" is used to describe technology (system, software, data) designed for a specific purpose and for a specific agency. Legacy systems are often proprietary, incompatible with external systems and are not designed to be extended by third-party developers.

Security Information Network, which was intended to act as a virtual gathering point for all levels of government. This chapter examines the information-sharing requirements HSIN was intended to fill, and the challenges facing development and adoption of a complex, government information sharing and collaboration system.

HSIN is a well-documented example of a major government information-system acquisition requiring the development of new software, integration of commercial and open-source software, and the need to connect to legacy system functionality and data across several agencies. Looking at how DHS managed this requirement provides lessons for future information-system management challenges.

A. HSIN DEVELOPMENT

DHS's original goal for HSIN was to provide a means to integrate information and communication services between federal, state, local, regional and tribal government entities in accordance with the Homeland Security Act of 2002[25]. Once complete, HSIN was to be "DHS's primary nationwide information-sharing and collaboration tool," incorporating data from all systems within DHS's jurisdiction.[26] The need to quickly establish a conduit between government agencies, to prevent another terrorist attack, pushed information system development to the top of many government agency's priority list. For DHS, an effective communication platform became one of the department's highest priorities[27].

[25] Homeland Security Information Sharing Act of 2002. Public Law 107–296, November 25, 2002, 11.

[26] Government Accountability Office, "Information Technology Numerous Federal Networks Used to Support Homeland Security Need to be Better Coordinated with Key State and Local Information-Sharing Initiatives: Report to the Chairman, Committee on Homeland Security, House of Representatives," 2007, 2, http://purl.access.gpo.gov/GPO/LPS82926.

[27] Ibid., 3.

By 2003, DHS planned HSIN to have thousands of initial users across sixteen government agencies. Early on, the decision was made to segregate users into communities based on their particular information and collaboration needs. Communities within HSIN were to be connected with an underlying emphasis on the entities they support within eight mission areas (see Table 1) that now total thirty-five communities of interest (COI). Each community, such as defense or law enforcement, has separate portals within HSIN.

Supporting Entities	Mission Focus
Federal	Critical Sectors
State and Local	Defense
Territorial	Emergency Management
Tribal	Homeland Security
	Intelligence
	Law Enforcement
	Multi-Mission
	International

Table 1. HSIN Entities and Community Focus Areas[28]

In early development stages, DHS identified eleven major legacy networks under its control that would have to be integrated into the HSIN framework. Existing systems like the Customs and Border Protection Network (CBP), Immigration, Customs Enforcement Network (ICENet), and Transportation and Security Administration Network (TSANet) are independently developed systems

[28] About Homeland Security Information Network. Department of Homeland Security (n.d.), http://www.dhs.gov/files/programs/gc_1156888108137.shtm.

comprised of proprietary software, data and communication protocols that presented a substantial development challenge in connecting these disparate systems. These challenges were compounded by a continuously accelerated HSIN delivery schedule and a lack of dedicated program management.[29]

B. INITIAL SYSTEM APPLICATIONS

The initial rollout of HSIN in 2004 consisted of four major component designed to connect government agencies and provide a conduit for sensitive but unclassified information. Web-based information sharing adopted by DHS was built around what then was called HSIN Enterprise Architecture 1.0, and consisted of multiple portal web pages, a discussion forum, real-time chat tools and a searchable document repository.

The purpose of the HSIN portal web application is to provide a framework for the delivery of user-specific content based on group membership.[30] Also referred to as a Content Management System (CMS) or dashboard, HSIN content page sections are delivered to users from multiple data sources populated with information based on COI membership, user configuration and assigned role. For example, an emergency management user's page contains generic homeland security-related sections, as well as pages, links and sections managed by the Emergency Management community. Portal pages also link to other HSIN resources that are tailored to a user's membership profile.

DHS discussion forums provide non-real-time, moderated, text-based discussions that also serve as an information repository for use by each community or interest. Forum posts are archived and searchable for use by

[29] Government Accountability Office, "Information Technology Homeland Security Information Network Needs to be Better Coordinated with Key State and Local Initiatives: Testimony Before the Subcommittee on Intelligence, Information Sharing and Terrorism Risk Assessment, Committee on Homeland Security, House of Representatives," 2007, 10, http://purl.access.gpo.gov/GPO/LPS83332.

[30] Web applications differ from a collection of static web pages in that delivered content can vary based on the a set of parameters and logic coded by the web developer. A Portal web application is a method to logically organize content sections on a webpage that is configurable by the user. Content sections typically contain links, news, email, and other data driven content.

current and future users as required. For example, a user may post a question, and then have multiple community members provide answers that are later accessible by COI users who have the same question.

The HSIN document library is a managed repository of regulations, directives and planning products, as well as user-submitted documents. The result is a searchable, continually growing archive of information segmented by community with many documents available via search requests available to all HSIN users.

While the other services mentioned provide a method to build collaborative knowledge over time, the HSIN's real-time chat component allows multiple users to discuss events as they happen. Text-based meetings between two or more geographically separated users can be conducted and archived for later use by meeting attendees.

Other system components included reporting and graphic applications that supply suspicious incident and pre-incident information, mapping and imagery, 24x7 situational awareness, and analysis of terrorist threats, tactics, and weapons.[31]

C. ROLLOUT AND RECEPTION

In February 2004, DHS officially launched HSIN portal with and its initial set of community-centric components. By July 2004, all fifty states and regional centers were connected and issued user accounts.[32] However, as the system was deployed at regional centers, it soon became clear that technical issues, limited or non-existent integration with legacy systems and duplication of existing functionality, would severely limit HSIN's acceptance.

[31] Government Accountability Office, "Homeland Security Opportunities Exist to Enhance Collaboration at 24/7 Operations Centers Staffed by Multiple DHS Agencies: Report to Congressional Requesters," 2006, 30, http://purl.access.gpo.gov/GPO/LPS76414.

[32] Relyea and Seifert. *Information Sharing for Homeland Security*, 5.

By the end of 2004, a flurry of GAO, Office of Management and Budget (OMB), and private organization reports critical of HSIN and other information-sharing initiatives began to surface. A 2004 Congressional Report to Congress (CRS) summed up the problem as

> … concerns about coordination and duplication of (government information sharing) initiatives have been raised since there currently appears to be no centralized inventory of all the information sharing initiatives being carried out within and between the federal, state, and local levels.[33]

In its rush to produce HSIN, it appears that DHS did not attempt to determine if similar systems were in use and did not develop an adequate set of user requirements.

The 2004 CRS report identifies four regional and national systems that provided similar functionality to HSIN and serve many of the same users. The rollout of HSIN resulted in confusion as to which system was primary for a given circumstance. For example, one issue concerned how law enforcement and emergency management systems would integrate with HSIN. The primarily law enforcement system called Regional Information Sharing System (RISS), provides identical functionality as HSIN. According to the GAO, HSIN program managers were unaware of the existence RISS during critical stages of HSIN's development:

> According to RISS program officials, they met with DHS twice (on September 25, 2003 and January 7, 2004) to demonstrate that their RISS ATIX application could be used by DHS for sharing homeland security information. However, communication from DHS on this topic stopped after these meetings, without explanation. According to DHS officials, they did not remember the meetings, which they attribute to the departure from DHS staff who had attended.[34]

---

[33] Relyea and Seifert. *Information Sharing for Homeland Security*, 5.

[34] Government Accountability Office. "Information Technology Numerous Federal Networks Used to Support Homeland Security Need to be Better Coordinated with Key State and Local Information-Sharing Initiatives: Report to the Chairman, Committee on Homeland Security, House of Representatives," 2007, 10.

D. REGIONAL INFORMATION SHARING AND HSIN

The Regional Information Sharing System is a Department of Justice (DOJ) information system designed to connect local, regional and federal law enforcement agencies and foster collaboration with other government agencies. Originally established in 1974, RISS is designed to first connect local agencies with regional centers and then to the national RISS network. "The RISS program uses a regional approach, so that each center can tailor/focus its resources on the specific needs of its area, while still coordinating and sharing information as one body for national-scope issues."[35]

RISS includes traditional web-based information-sharing applications like web portals for each region, forums, real-time chat and a document library, as well as applications focusing on law enforcement issues. Application databases include RISSGang, for collecting and sharing information related to gang activity, and RISSIntel, for the collection and search of crime-based intelligence.

RISS anti-terrorism initiatives include the Anti-Terrorism Information Exchange (ATIX) system. This component of RISS was developed in 2002 to "facilitate communication and information sharing among personnel responsible for planning and implementing actions to prevent, mitigate, and recover from terrorist incidents and disasters." In fact, ATIX was a key player for law enforcement and disaster recovery efforts following Hurricane Isabel in September 2003. ATIX was also the primary communication and planning mechanism for the 2004 G8 Summit in Georgia and the Republican and Democratic conventions.[36]

In DHS's rush to create an information sharing network for its perspective users, a survey of existing systems was never adequately conducted. The GAO reported in 2007 that HSIN was developed and deployed without an "understanding of the relevance of the Regional Information Sharing Systems

[35] Relyea and Seifert. *Information Sharing for Homeland Security*, 9.

[36] Ibid., 10.

program to homeland security information sharing."[37] The result was a duplicate system targeting the same users with nearly identical features.

Today, DOJ agencies continue to use RISS primarily for information sharing and HSIN for monitoring real-time incidence like natural disasters. During national events like a presidential inauguration, control centers must monitor multiple systems and, today, still have no easy way to manage these multiple duplicative systems.

The challenge for HSIN is to integrate RISS components and data into DHS-based systems to, at a minimum, allow users to search RISS-based crime data. So far, HSIN program managers have been unable to incorporate RISS data due to architectural limitations inherent in HSIN underlying structure—highlighting both the need for a flexible, interoperable architecture (described in Chapter V) as well as effective program management practices.[38]

E. HSIN NEXT GENERATION DEVELOPMENT

By the beginning of 2008, DHS had poured over $90 million into HSIN and over $611 million combined into the eleven homeland security networks under its control.[39] Despite seemingly adequate funding and over four years of development, HSIN still had few users; DHS stopped actively marketing the system since it was clear that HSIN was not meeting DHS' own expectations.

Early in 2008, DHS decided to scrap further development of HSIN and begin planning a revamped version of HSIN. The new system, called HSIN Next

---

[37] Government Accountability Office. "Information Technology: Homeland Security Information Network Needs to be Better Coordinated with Key State and Local Initiatives: Testimony Before the Subcommittee on Intelligence, Information Sharing and Terrorism Risk Assessment, Committee on Homeland Security, House of Representatives," 2.

[38] Government Accountability Office. "Homeland Security Efforts Under Way to Develop Enterprise Architecture, But Much Work Remains: Report to the Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Committee on Government Reform, House of Representatives," 2004, 1.

[39] Government Accountability Office. "Information Technology: Numerous Federal Networks Used to Support Homeland Security Need to be Better Coordinated with Key State and Local Information-Sharing Initiatives: Report to the Chairman, Committee on Homeland Security, House of Representatives," 1.

Generation or “Next Gen,” was to be an architectural redesign that would more easily incorporate data and functionality from external sources and third-party applications. Acknowledging past management failures, project improvements for Next Generation include the establishment of a program management office to oversee system and acquisition planning. DHS established a full-time project manager (PM) and staff for the first time since HSIN development began. Included with the full-time PM team is a staff position responsible for gathering requirements from all users and for surveying existing systems for HSIN integration of legacy data and functionality.

The DHS also established a Homeland Security Information Network Advisory Council (HSINAC) with the mission to improve the effectiveness of HSIN information-sharing initiatives, and a mandate to oversee the development and improvement of the next generation system. HSINAC held the first annual three-day meeting in late October 2007. This meeting established policy, business process and governance requirements needed to better manage HSIN.[40]

Since its first meeting, HSINAC has helped create a comprehensive HSIN governance structure to ensure system requirements “are directly tied to mission areas and communication capabilities.”[41] Figure 1 provides clues to how the advisory council has improved HSIN program management and its attempt to match program capabilities with system requirements. As the Change and Improvement Flow chart shows (Figure 1), DHS Operations organization flows results of user outreach initiatives into business requirements that are then considered by a Change Control Board (CCB) chaired by the HSIN program

[40] Meeting Minutes: Homeland Security Information Network Advisory Committee Inaugural Meeting, October 30 – November 1, 2007, *The Department of Homeland Security*, December 28, 2007.

[41] Ibid., 2.

manager. Approved system changes, improvements, and/or additions are then packaged for implementation during upgrades. Under this new paradigm, user requirements are the primary driving force for change.

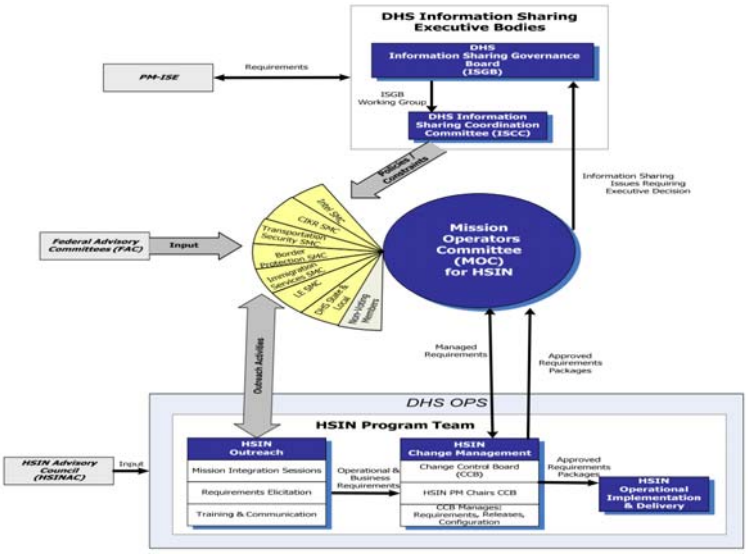

Figure 1. HSIN Change and Improvement Flow[42]

F. HSIN NEXT GENERATION APPLICATIONS

Improved management and user outreach, along with the funding of an enhanced architectural design, has allowed HSIN Next Generation to improve integration of contract-developed applications, commercial components, and open source software. The Appendix provides a description of how these new components were used during the recent Deepwater Horizon disaster in the Gulf of Mexico.

1. HSIN Common Operating Picture

The HSIN portal includes a Common Operating Picture (COP) component similar to systems used by the Department of Defense. HSIN COP provides

[42] Final Report: Homeland Security Information Network Advisory Committee Meeting, February 10–12, 2009, *The Department of Homeland Security*, March 27, 2009, 11, http://www.dhs.gov/xlibrary/assets/hsinac_mtg_2009-2-1012.pdf.

users with real-time, constantly updated information concerning new and ongoing DHS operations. The COP also tracks media reports and internal and external requests for information. Task responsibility is assigned for each information request along with up-to-the-minute status information accessible to all portal users. Each task has a tracking date, source, and resolution information once complete.

2. HSIN Connect

Representing a huge advance in real-time collaboration for HSIN, Connect is a virtual meeting tool available to all system users for anytime, on-demand, online video meetings via point-to-point encrypted data for enhanced security. Users can view documents on the presenter's screen during the session and record the session for later viewing. Connect is made by Adobe Systems, and is one of the first third-party tools incorporated into HSIN.

3. Wikis and Online Reading Rooms (Open Source Component Integration)

HSIN Next Generation's improved architecture design now includes a provision to integrate open source components. An example is the establishment of general information wikis as well as online reading rooms created to ensure all users have the latest information concerning HSIN-tracked events.[43] Users with edit privileges can add, delete, or correct information on the fly. One particularly useful section is a *lessons learned* wiki. From these pages, user can create additional wiki pages where users can post questions, make comments or request additional information.

[43] A wiki is an open source web-based collaboration tool that allows users to create content for others to add, delete, or modify as required. For additional information, visit en.wikipedia.org/wiki.

4. Federated Search and Role-Based Data Access

One of the original purposes of HSIN was to facilitate information sharing between government agencies. As discussed in this section, a lack of management, planning, user outreach and surveys of existing systems during HSIN's initial planning resulted in an inflexible architecture that limited the integration of external data and functionality. The difficulty stems from database compatibility and connection issues between incompatible data sources and communication protocols. To solve compatibility issues, a recent search paradigm, Federated Search, allows separate systems to feed external data requests without the need for a huge centralized database. Federated search, along with strict role-based data access, will eventually allow an HSIN user to search disparate data sources with results tailored to user type or community. "This means that a Sherriff Doctor with a Secret clearance can see all the law enforcement, medical, and secret information."[44] Federated search is enhanced by component and Service Oriented Architectural design practices discussed in Chapter V of this thesis.

G. LESSONS FROM HSIN

The planning problems DHS encountered developing and deploying HSIN are common in large-scale software acquisition, with "many projects, perhaps 20 percent, will be abandoned, often after multimillion-dollar investments—and the biggest projects will fail most often."[45] Pressure to rapidly develop and deliver a fully functioning, multiple-user system across several government agencies will almost certainly be encountered again in the future. These projects may be large, national-level system like HSIN, or smaller initiatives like Intellipedia (detailed in Chapter IV). Regardless of size, these new systems will almost

[44] Final Report: Homeland Security Information Network Advisory Committee Meeting, February 10–12, 2009, 18.

[45] Software Engineering Institute, Carnegie Mellon: Software Development, 2010, http://www.sei.cmu.edu/solutions/softwaredev/.

certainly require a combination of internally developed technology, off-the-shelf software and custom-built components, along with the use of open source software to satisfy a set of requirements. To improve these future projects, it is beneficial to look at large-scale system acquisitions like HSIN to help mitigate the risk of repeating past mistakes.

One of the main shortcomings noted in nearly every report concerning the development of HSIN was a lack of program management early in the project. It is evident in these reports that operational necessity trumped the need for solid management practices. It can be argued that DHS's initial and almost fatal mistakes were in not creating a full-time project manager with an adequately staffed office during the initial planning and design phases of the project. This lack of management led to the following missteps during early HSIN development and deployment:

1. Inadequate Requirement Planning and Management

Understanding the problem a new system is trying to solve and how best to meet user's needs is fundamental to system design planning. Inadequate requirement planning caused HSIN to have an increased risk of exceeding "project costs, delayed schedules and performance shortfalls."[46] As the GAO noted, initial HSIN planners did not adequately survey potential users or attempt to determine what government systems are currently in use by HSIN's intended user base. The next chapter of this thesis focuses on best practices in system requirement planning that could have helped HSIN during early planning and are likely beneficial for future government system acquisitions.

[46] Government Accountability Office, "Information Technology Management Improvements Needed on the Department of Homeland Security's Next Generation Information Sharing System: Report to Congressional Requesters," 2008, 14, http://purl.access.gpo.gov/GPO/LPS104962.

2. Inadequate Risk Planning and Management

Risk management during a project's life cycle helps ensure potential problems are managed and (when possible) mitigated. To some planners, project risk management is a luxury that time often does not permit. However, in many cases, good risk management practices help identify design flaws that could potentially result in schedule and cost overruns. For HSIN, risk planning did not begin until the development of the next generation system; "however, they (HSIN risk managers) have yet to identify all key risks surrounding the project and develop risk mitigation plans."[47]

Chapter V of this thesis identifies SEI, DACS and other leading government acquisition resource best practices for software risk management. These software risk-planning resources show that devoting time to risk management helps planners anticipate and effectively react to problems that can lead to cost and schedule overruns.

3. Inadequate Architectural Design Practices

Selecting an architectural design is fundamental to the success of complex systems like HSIN. Poor system design and planning can lead to a chaotic mix of functionality that is difficult to maintain and secure. Modern design practices incorporate component architectures that break complex projects into manageable pieces. Chapter V of this thesis examines best practices in system architecture that fosters the integration of legacy systems that easily supports future technology.

H. CONCLUSION

Since HSIN was first delivered in 2004, the system has struggled to meet its user's need for a single source of homeland security information and

[47] Government Accountability Office, "Information Technology Management Improvements Needed on the Department of Homeland Security's Next Generation Information Sharing System: Report to Congressional Requesters," 2008.

collaboration. Fortunately, recent improvements in architecture, integration of commercial components and a dedicated program management team have helped HSIN Next Generations gain acceptance within the DHS community.

To help HSIN and other government system acquisition programs, the GAO, CMS and non-governmental organizations have identified the source of HSIN's initial development shortcomings. These include not assigning a full-time project management team, which ultimately led to inadequate requirements planning, risk management and architectural design practices. The remainder of this thesis is devoted to identifying key best practice in each of these areas of software system development, as well as acquisition resources useful for further study.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPORTANCE OF ACQUISITION PLANNING AND REQUIREMENTS MANAGEMENT

> Question: What does the timing look like (to provide better data interoperability for HSIN)?
>
> Answer: This could take five years…HSIN itself was put out under threat conditions, there wasn't time to lay out the plan–due to operational necessity.
>
> —HSINAC meeting minutes, February 12, 2009[48]

Effective acquisition of government information-sharing systems is critical to the success of agencies that are mandated is to piece together disparate data streams and combine them into actionable information. Developing an acquisition strategy based on lessons learned from past development, commercial industry standards and academia, along with advances in component architectures that foster interoperability and extensibility, can help increase the probability of success for future acquisition projects.[49]

Government and industry software and systems development is a vast subject area to consider. A project manager—with a mandate to oversee systems that combine multiple external data sources, have thousands of role-based users, contain sensitive information, and integrate internally developed, commercial, purchased and open-source software—has literally thousands of high-level details to consider. The scope and scale of such an undertaking is the subject of countless volumes of research. Despite this complexity, it is worth attempting to identify key issues that hamper the success of government system projects and locate solutions that reduce complex problems into manageable subsets.

---

[48] Final Report: Homeland Security Information Network Advisory Committee Meeting, February 10–12, 2009, 18.

[49] Extensibility is a software design feature that allows functionality to be added in the future. This helps insure software systems do not become obsolete as new technology is developed. Interoperability allows software to exchange information with internal or external systems.

This chapter begins with a look at why large acquisition projects fail and how lessons learned from failed projects can be used to build better project management practices. This chapter also considers system project planning and how carefully developed requirements are critical to the success of an effective software acquisition strategy. These requirement processes continue throughout a program's life cycle and feed into the risk and architectural strategies covered in Chapter V.

A. WHY ACQUISITIONS SOMETIMES FAIL

Statistics from Carnegie Melon's Software Engineering Institute portal show that HSIN is not unique in its struggle to evolve from initial development to a delivered, user-accepted product:

> Organizations and governments worldwide will spend about $1 trillion this year on IT projects. Recent data suggested only about 35 percent of those projects are likely to be completed on time and on budget, with all their originally specified features and functions. Many projects, perhaps 20 percent, will be abandoned, often after multimillion-dollar investments—and the biggest projects will fail most often.[50]

Attempting to understand why software acquisition projects fail is a well-established area of research. Over the past twenty years, lead sources for government program improvement include the Defense Science Board (DSB), Software Engineering Institute (SEI) and the National Defense Industrial Association (NDIA), as well as the GAO. In fact, the GAO's 2004 Defense Acquisitions reports are considered a "rallying point for any and all acquisition organizations who are struggling to improve the results from their software acquisition processes."[51] Also, a recent Carnegie Melon academic study, "Lessons Learned from a Large, Multi-Segment, Software-Intensive System," is

[50] Software Engineering Institute, Carnegie Mellon: Software Development.

[51] The Data and Analysis Center for Software: Software Acquisition Gold Practice, 2010, https://goldpractice.thedacs.com/practices/api/.

devoted to the shortcomings commonly found in government acquisition projects and is the culmination of many years of research.[52]

According to these sources, many of the reasons for information technology (IT) project failures are also areas identified as shortcomings associated with HSIN's early development. According to these sources, some of the most common reasons information technology (IT) acquisitions fail are a lack of initial requirements, risk and architectural planning. In fact, according to the GAO and DHS's own HSINAC, a lack of initial planning and research are the main reason HSIN failed to attract users.[53] These primary sources of acquisition research indicate that it is critical for managers to understand the importance of these areas of software acquisition planning.

Over the years, since DHS first deployed HSIN, it became clear that HSIN's "initial development was not based upon a solid set of user requirements; as a result, the performance of HSIN program management was not adequate."[54] The rush to develop and deploy a system—any system—lacked the management oversight needed to create a coherent plan. DHS subsequently found that pushing an inadequately planned and managed system onto their user base created a difficult set of challenges. DHS is now painfully aware that fixing an existing flawed system is much more difficult than doing the necessary upfront planning. DHS also found that marketing an ineffective system hampers its adoption and use, even as system improvements are later delivered.[55]

Fortunately, DHS's establishment of an HSIN program office, with a full-time program manager and staff in 2007, helped the development of the next

---

[52] Lessons Learned from a Large, Multi-Segment, Software-Intensive System, *Software Engineering Institute, Carnegie Mellon*, n.d., 1, http://www.sei.cmu.edu/library/abstracts/reports/09tn013.cfm.

[53] Final Report: Homeland Security Information Network Advisory Committee Meeting, February 10–12, 2009, 18.

[54] Ibid., 3.

[55] HSINAC Committee's annual meeting minutes from March 2007 and March 2009 detail the difficulty in marketing system improvements.

generation system currently being deployed. However, shortcomings associated with HSIN's early development serve as an anchor that demonstrates the need for management practices that include requirement, risk and architectural planning. If DHS had initially established a full-time program manager, with adequate time to lay a solid foundation in these three areas, HSIN usability and acceptance would certainly look different today.

B. PROJECT REQUIREMENTS PLANNING

Managing the development of a large, complex system requires a plan. The fundamental focus of a project plan is to clearly lay out the problem the system is trying to solve. Understanding and documenting the core problem set allows the program manager to define and communicate the minimum essential high-level tasks that must be accomplished during system development.[56] Clearly establishing these minimum requirements helps focus development team-member activities and helps to defend against the pitfalls associated with inadequate planning.

A comprehensive case study of fifteen successful software-system development projects, published in IEEE Software Journal in conjunction with the Department of Defense Information Analysis Center, identified several key requirement practices used in successful project acquisitions.[57] Software acquisition best practices identified by this study separate requirements planning into subtasks that include user elicitation, requirement analysis, modeling and validation. These subtasks occur throughout development, and continue as the system is deployed and upgraded. Once initial requirements are established, timelines, benchmarks and milestones form the pillars of a management plan that guides the entire process.

[56] The Data and Analysis Center for Software, "Requirements Engineering," n.d., https://www.thedacs.com/databases/url/key/5086.

[57] Ibid.

1. User Requirements Elicitation

Understanding user needs is a critical component of requirements planning that continues throughout the development process. "The most successful teams always involve customers and users in the requirements elicitation (RE) process…according to one study, user participation is one of the most important factors contributing to requirements engineering success."[58]

For HSIN, meeting user requirements is one of the most frequently cited areas needing improvement.[59] Clues to fixing this requirements task can be found in SEI's Capability Maturity Model Integration (CMMI), considered the software systems acquisition *bible* by many in government and industry. CMMI Software Goal 1 (SG-1) states that user needs, expectations, and interfaces must be translated into a concise document that evolves during a product development. SG-1 recommends that the project management team observe user workflow patterns, and conduct interviews and operational scenarios to determine the technical functionality required by users. The user "typically describes requirements as capabilities expressed in broad operational terms concerned with achieving a desired effect under specified standards and regulations."[60] The effect described should also have enough detail to guide user interface designs. For example, instead of simply indicating the user would like database search functionality, the requirement should also describe filtering, content needs and a description of how the user interface should display the output. The solicitation process should continue through all phases of development and deployment to ensure current and future needs are included in the process.

---

[58] Hubert F. Hofmann, "Requirements Engineering as a Success Factor in Software Projects," *IEEE Software*, July 1, 2001, 65.

[59] Government Accountability Office, "Information Technology Management Improvements Needed on the Department of Homeland Security's Next Generation Information Sharing System Report to Congressional Requesters," 3.

[60] *CMMI for acquisition, Version 1.2: CMMI-ACQ, v1.2.* (Pittsburgh, Pa.: Carnegie Mellon University, Software Engineering Institute, 2007), 97.

Once a comprehensive list of requirements is developed, CMMI recommends prioritizing the list to ensure user priorities also meet the needs of the organization. Prioritized requirements are then analyzed and modeled in order to build milestones and performance goals.

User and Stakeholder inputs (examples)	**Intermediate Process**	**User Requirement Outputs**
User Questionnaires	Compile list based on user inputs	Prioritized User Requirements (to feed program definition, analysis and validation processes)
Discussion Groups	Resolve conflicting requirements	
Operational scenarios from end users	Prioritize requirements list	
Internal business process documents, standards or specifications	Consider potential obstacles, supportability	

Table 2. Sample Requirements Inputs, Processes and Outputs[61]

2. Analysis and Modeling

Both the CMMI and IEEE acquisition documents recommend building models that analyze user and stakeholder requirements in terms of minimum operational needs of the proposed system. The analysis phase is simply a further refinement of user-solicited requirements that are later matched to functionality. For example, some user-gathered inputs are valid, but do not fit overall functional requirements needed for initial operations. Below the line requirements contain “nice to have” functionality to include if resources permit, but are not necessary for the system to be considered functional. Other aspects of analysis concern stakeholder needs that addresses proposed functionality in

[61] *CMMI for acquisition, Version 1.2: CMMI-ACQ, v1.2.* (Pittsburgh, Pa.: Carnegie Mellon University, Software Engineering Institute, 2007), 96.

terms of "cost, schedule, performance, functionality, reusable components, maintainability, and risk."[62] Chapter V of this thesis covers the latter three architectural-related tasks.

Modeling is a relatively new concept that involves creating interface prototypes that allow developers to simulate proposed minimum functionality based on the user requirement document developed during requirements analysis. These models can serve to synchronize developer, stakeholder and user understanding of how the interface should deliver functionality that matches identified requirements. Models can use custom-built simulations or can be demonstrated using existing applications whose functionality will be combined to form the user interface.

3. Validation and Verification

Validation and verification is essentially a big-picture sanity check conducted after the initial requirements document is complete. The stakeholders and developers work together to ensure requirements are properly prioritized, that requirements included in the initial design meet minimum essential system needs, and that lower priority items can be integrated in later versions if necessary. Again, these decisions feed the risk and architectural design requirements conducted later in the development cycle.[63]

C. REQUIREMENTS PROGRESS MANAGEMENT

Once initial operating requirements are established through elicitation, analysis and modeling, the IEEE recommends selecting processes that will serve as progress indicators.[64] These indicators establish core task milestones that are integrated into progress and management review timelines that facilitate

---

[62] *CMMI for acquisition, Version 1.2: CMMI-ACQ, v1.2.,* 104.

[63] IEEE Computer Society. Software Engineering Standards Committee. and Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Software Acquisition*, 59.

[64] Hubert F. Hofmann, "Requirements Engineering as a Success Factor in Software Projects," 62.

resource and allocation planning. Without a program-managed set of core tasks, development often strays as complexity increases and user requirements evolve during development. Constant unchecked changes without milestones can lead to time and cost overruns.[65] Changes during the development life cycle are necessary and even desired; however, without a mechanism to manage alterations to initially agreed-to functionality, projects can grow out of control. DACS gold practices for project management states it this way:

> Requirements Management (RM) seeks to reduce the risk of cost and schedule overruns by establishing a way to control the continuing definition of requirements as changes occur and unforeseen needs arise and as knowledge is gained during development, in contrast to more traditional development approaches where requirements were documented (often without involvement of the developer) prior to any development activities and frozen for the life of the effort. The successful implementation of RM depends on having flexible scenarios that require the establishment of a process to manage requirements (in lieu of rigid pre-defined specifications) that addresses specification, change control and traceability, and identifies what stakeholders must be involved in the various activities of the process throughout the life cycle.[66]

This evolutionary approach allows the PM team to manage requirement changes without the project losing focus on the core problem set discussed earlier.

D. SURVEY OF EXISTING TECHNOLOGY

Designing and managing large-scale systems that incorporate data from multiple legacy systems, replace and/or add functionality and include commercial and open-source components requires intensive upfront and ongoing planning to establish a flexible and interoperable architecture. However, before selecting an architecture, it is important to understand what part of the system must be developed in-house, and can be commercially purchased or acquired as an

[65] The Data and Analysis Center for Software, “Requirements Engineering.”

open-source solution that is later modified to meet previously defined requirements. The CIA's information-sharing program, Intellipedia, described next, provides a good example of a government organization matching user requirements with existing functionality.

E. CASE STUDY: INTELLIPEDIA AS AN OPEN SOURCE SOLUTIONS FOR GOVERNMENT INFORMATION SHARING

In 2006, CIA officer Calvin Andrus wrote an essay concerning government information sharing and the Internet titled, "The Wiki and the Blog: Toward a Complex Adaptive Intelligence Community."[67] In his essay, Andrus defined the problem he sought to solve by first arguing that information management techniques must evolve in order to be useful to the end user. Intelligence managed by a small subset of information managers attempting to maintain content from thousands of sources is inherently inefficient. Seemingly insignificant pieces of data that could be tied together to form useful intelligence often slips through the cracks when large amounts of data pass through few hands.

To solve this problem, Andrus suggested that individual intelligence officers be empowered to shape source data in real time and "be allowed to react—in independent, self-organized ways—to developments in the national security environment."[68] According to Andrus, intelligence data must also be easily shared with all users and include a mechanism for feedback from anyone in the community.

Andrus' call, for more dynamic, independent and self-organized information sharing that is less centrally managed and more user accessible, is a good starting point for a software acquisition manager to transition from a defined

---

[66] The Data and Analysis Center for Software, "Requirements Engineering."

[67] D. Calvin Andrus, The Wiki and the Blog: Toward a Complex Adaptive Intelligence Community, Studies in Intelligence, 2005. http://ssrn.com/abstract=755904.

[68] Ibid., 3.

problem to a user requirement. In this case, a requirements list could be developed that included interface requirements describing how data should be input, edited and commented on by users.

Fortunately, for this particular problem set, a model already existed, one that met the user's need for a user-edited content-management system. The open-source web-based content system, called a Wiki, allows individuals to "self-organize around shared knowledge."[69] Once set up, the Wiki interface allows information contributors to add information, edit other user's information and provide amplifying comments anywhere within the document. Any user can create a new document that the community can edit as the situation evolves.

In this case, the model (Wiki software) fits the requirements so closely that the model itself becomes the solution. The intelligence-inspired Wiki became known as Intellipedia and is in widespread use in the intelligence community today. The selection of an open-source, easily managed system almost certainly reduced the cost of acquiring a system to meet the user requirements established by Andres' essay.

F. CONCLUSION

Understanding why major acquisition projects fail can help future project managers create strategies to avoid common pitfalls that have plagued past acquisition projects. Organizations like the GAO and Carnegie Melon's SEI have identified requirements, risk and architectural design as planning areas commonly neglected in struggling and failed government software acquisitions. These planning areas are also cited by GAO and CRS reports as weaknesses associated with HSIN's initial development.

A basic task for any new software system is to determine the problem it is trying to solve. Best practices in project planning help program managers develop, prioritize, analyze and model requirements in order to ensure proposed

[69] D. Calvin Andrus, The Wiki and the Blog: Toward a Complex Adaptive Intelligence Community, Studies in Intelligence, 2005, 3. http://ssrn.com/abstract=755904.

functionality meets the user's needs. Good requirements planning also helps establish minimum initial functionality and milestones that keep the project from exceeding budget and time limitations. A solid understanding of the user's needs and matching functionality can then be analyzed for developmental risks and broken into manageable components that lay a foundation for selecting an appropriate architecture.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RISK MANAGEMENT AND ARCHITECTURAL STANDARDS FOR COMPONENT INTEGRATION

I have not failed. I've just found 10,000 ways that won't work.

—Thomas Edison

Managing a large project certainly involves some degree of risk. Whether risk ultimately results in failure often depends on preparation and planning for the unexpected. Chapter IV discussed requirements as the first of the three commonly neglected areas of software system acquisition (requirements, risk, architecture planning). As described in Chapter IV, best practices in requirements management include elicitation, analysis, modeling and validation to help match user needs with application functionality. During this process, requirements can be matched to a model to help the developer and stakeholder fully define the problem as well as find potential solutions through contract-developed applications, commercial off-the-shelf software (COTS) and/or open-source projects. From here, complex functionality can be separated components that, when combined, comprise an initial system design.

How these components interact, and how data is shared and distributed, is part of the system architectural design process that involves decisions that have ramifications throughout the projects life cycle. Implementing the wrong architectural solution can mean legacy and newly created systems do not easily interact or share data and functionality. Architecture, together with requirement planning, ultimately allows project managers to begin to build a comprehensive project plan. However, plans developed during this stage almost always encounter obstacles during execution. Devoting time to plan for potential obstacles is key to mitigating the risk of a project going off track.

This chapter examines two often-overlooked areas of system design that should be considered from the beginning of the system acquisition process: risk management and architectural design. Planning for risk is sometimes seen as a

nice to have but not crucial aspect of system planning. However, good risk management can help identify design weakness that later result in delays and cost overruns. System architecture is another planning task sometimes given minimal time and resources. However, systems that are haphazardly pieced together components can lead to unintended consequence that include dependence on proprietary data and code, security flaws and interoperability issues—all avoidable with good architectural planning.

A. SOFTWARE RISK MANAGEMENT

1. Defining Software Development Risk

With an adequately researched and prioritized set of user requirements and procedures in place to manage these requirements, the project manager should next consider risks that can cause project delays. The Data and Analysis Center for Software defines risk as:

> A proactive approach for minimizing the uncertainty and potential loss associated with a project. A risk is an event or condition that, if it occurs, has a positive or negative effect on a project's objectives. Future events can be categorized as opportunity-focused (positive risk) if their consequences are favorable, or as threat-focused (negative risk) if their consequences are unfavorable.[70]

At a basic level, software risk management is a process for developing a list of hazards or problems that could reasonably occur through a system's life cycle, determine the probability of each occurring, and develop plans to mitigate or otherwise react to negative events. Failing to develop and manage a risk plan can be likened to failing to purchase auto insurance. If the insurance purchased is never needed, it is tempting to consider the expense a waste of money. However, if an accident does occur, insurance can prevent financial ruin, and is therefore seen as a positive.

[70] The Data and Analysis Center for Software, Risk Management, 2010.

Setting aside development time specifically for risk management, particularly for projects with limited development time, can seem unnecessary. However, over the past fifteen to twenty years, software complexity has grown exponentially. Without an adequate risk plan and management process, program managers can find themselves fighting fires instead of effectively managing the unexpected.

> There are two ways of dealing with risk. One, risk management, is proactive and carefully analyzes future project events and past projects to identify potential risks. Once risks are identified, they are dealt with by taking measures to reduce their probability or to reduce their impact. The alternative to risk management is crisis management. It is a reactive and resource-intensive process, with available options constrained or restricted by events.[71]

When risk is not carefully considered and the unexpected delay does occur, risk planning is correctly highlighted as lacking. For HSIN, risk management has been cited as one of three major factors contributing to the systems shortcomings. In 2008, the GAO reported that HSIN's aggressive upgrade schedule had precluded adequate risk management planning. In fact, it was not until five years into the development of HSIN that the implementation of a risk management plan was established. As noted in a 2008 GAO report, "DHS has begun to develop a risk management plan that defines staff roles and responsibilities. However, it has yet to identify all key risks surrounding the project and develop risk mitigation plans and completion milestones."[72] As illustrated next, adding risk planning is a straightforward process that continues through the project's life cycle.

---

[71] Software Technology Support Center, "Understanding Risk Management," *CrossTalk*, 2005, http://www.stsc.hill.af.mil/crosstalk/2005/02/0502stsc.html.

[72] Government Accountability Office, "Information Technology Management Improvements Needed on the Department of Homeland Security's Next Generation Information Sharing System: Report to Congressional Requesters," 2008, 4.

2. Risk Methodologies

Systematic methods to manage risk in software development date back to the late 1980s with the IEEE's tutorial "Software Risk Management," in which core risk concepts were established.[73] In this paper, Dr. Barry Boehm defines the purpose and importance of risk management planning as methods to:[74]

1. Avoid software project disasters, including runaway budgets and schedules, defect-ridden software products, and operational failures.

2. Avoid rework caused by erroneous, missing, or ambiguous requirements, design or code, which typically consumes 40–50% of the total cost of software development.

3. Avoid overkill with detection and prevention techniques in areas of minimal or no risk.

4. Stimulate a win-win software solution where the customer receives the product they need and the vendor makes the profits they expect.

Boehm makes the point that risk management, regardless of project type, is a continuous cycle of risk analysis, prioritization and planning that highlights potential problems and provides contingency plans for in case those problems arise. Without a risk plan, problems are simply address as they occur, which almost certainly increase costs and results in project delays as solutions are sought on the fly.

While specific risk management processes vary for each project, Figure 2 provides a good illustration of a generic project risk flow. Planning begins by identifying and analyzing problems that have the potential to compromise a project's success. Identifying risks requires time to be set aside specifically for

[73] Barry W. Boehm and Ez. Nahouraii, IEEE Computer Society, *Software Risk Management: Principles and Practices* (IEEE Computer Society Press, 1989).

[74] Ibid., 89.

stakeholders, requirements managers, developers and project manager to brainstorm the challenges that could present themselves during the projects development.

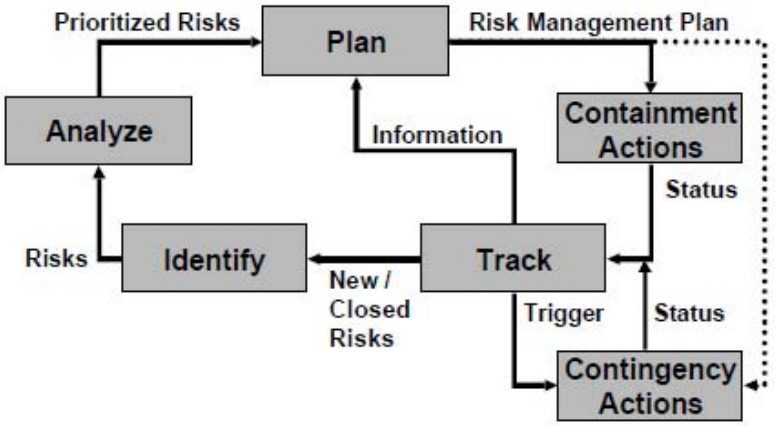

Figure 2. Risk Management Process[75]

Once risks are identified, project planers should then devote time to examining the likelihood each identified risk has of occurring. In order to rack-and-stack identified risk, Boehm offers a simple risk equation that allows planners to quantify risk:

RE = Probability (UO) * Loss (UO), Where UO = Unexpected Outcome

Calculating probability and loss are subjective and often rely on experience gained from past project acquisition and development. Boehm's method forces planners to consider how likely an event will occur (Risk #1) and if identified risks could increase in probability of another identified risk occurring (Risk #2). This process helps prioritize the list of risks so that resources can be appropriately allocated and continuously tracked and reassessed.

[75] Linda Westfall, "Software Risk Management," *The Westfall Team*, 2001, http://www.westfallteam.com/Papers/risk_management_paper.pdf.

Since the original work was published, the SEI and IEEE have expanded upon Boehm's work by developing software risk-management frameworks that simplify incorporating risk planning into any project.[76] Also, risk checklists from NASA and SEI, as well as non-software-specific risk checklists from Arizona State University and the Department of Energy, are considered industry standards.[77]

3. Advances in Risk: From Tactical Risk to MOSAIC

As government software projects increase in complexity, the need for initial and ongoing detailed planning becomes increasingly critical. While the IEEE and SEI have advanced the field of risk planning for government systems, the potential for failure increases as systems become more complex. To meet these challenges, SEI has expanded its basic software risk framework, which "codified" risk management best practices, and laid a foundation for further advances.[78] The first such advance is their Mission-Oriented Success Analysis and Improvement Criteria (MOSAIC) process, intended to help project managers maintain control of large, distributed system development that is increasingly common in government settings.[79]

The main advance MOSAIC provides is a shift from tactical risk planning to a higher-level framework, better suited to a distributed development environment. SEI describes the traditional tactical risk approach as planners looking for what can go wrong, determining which of these risks are most

---

[76] Christopher J. Alberts and Audrey J. Dorofee, *Risk Management Framework* (Software Engineering Institute, Carnegie Mellon, 2010), http://www.sei.cmu.edu/reports/10tr017.pdf.

[77] Arizona State University, *Question List for Software Risk Identification in the Classroom*, n.d., http://www.eas.asu.edu/~riskmgmt/qlist.html.

[78] Christopher J. Alberts and Audrey J. Dorofee, *Risk Management Framework* (Software Engineering Institute, Carnegie Mellon, 2010), http://www.sei.cmu.edu/reports/10tr017.pdf.

[79] Distributed systems are application and hardware infrastructures that connect multiple networked computers that form clusters that connect to other clusters. Each cluster can have a separate user-base and purpose. Applications must communicate within clusters and with other clusters. Managing distributed system can create complex communications, reliability, availability, serviceability and scalability issues that require innovative risk management techniques.

important and allocating resources on the most likely risks identified.[80] This approach has worked well for systems that operate with few interconnections, like desktop applications for example, but are not as well suited for complex interconnected system development.

Networked distributed systems like HSIN operate in a dynamic environment with multiple layers of separately developed legacy systems that have a high degree of uncertainty when considering potential risks. A bottom-up tactical risk analysis typically focuses on corrective action associated with each identified risk occurrence, but does not adequately address the impact of a risk's consequences on the network of systems. This is because distributed system development contains such a large number of risk variables that predicting the outcome of a particular event becomes increasingly difficult.

The intent of SEI's MOSAIC is to solve the shortcomings of the traditional tactical risk planning by analyzing a project in terms of its processes. Processes have drivers that "guide the outcome (of a process) toward key objective (success state) or away from them (failure state)."[81] SEI identifies twenty drivers associated with software system development; they range from defining program objectives, and planning to final certification and acceptance.

[80] Audrey Dorofee and Christopher Alberts, Rethinking Risk Management: NDIA Systems Engineering Conference. Software Engineering Institute. 2009.

[81] Ibid., 32.

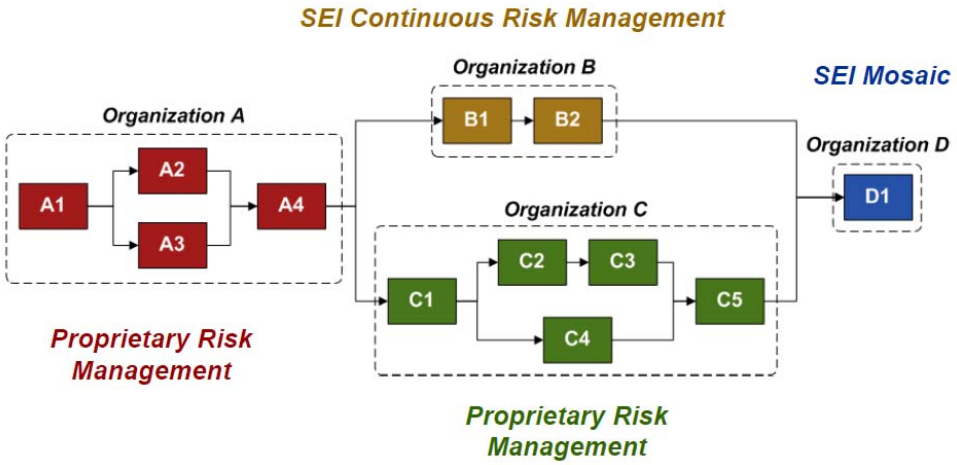

Figure 3. Risk Management MOSAIC for Multi-Enterprise Environments[82]

Drivers are divided into categories that cover the spectrum of the software development life cycle that can be analyzed across an organization. As shown in Figure 3, organizational analysis is combined for use in system-wide analysis and planning across segmented proprietary organizations using legacy systems. This approach helps planners visualize risk interaction across system segments to determine how individual risks affect the broader enterprise. This approach is easier to accomplish when the system is comprised of components organized by a component-based architecture, as discussed next.

B. SOFTWARE ARCHITECTURE

Another aspect of software system development sometimes overlooked in time-critical projects is architectural design. Software architecture is defined by IEEE Standard 610.12-1990 as "the structure of the components of a program and/or system, their interrelationships, and principles and guidelines governing

[82] Dorofee and Alberts, Rethinking Risk Management: NDIA Systems Engineering Conference. Software Engineering Institute, 2009, 129.

their design and evolution over time."[83] Another useful definition concerns the structural relationship between components and their effect on managing risk:

> Software architecture of a computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. By 'externally visible' properties, we are referring to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. The intent of this definition is that a software architecture must abstract away some information from the system (otherwise there is no point looking at the architecture, we are simply viewing the entire system) and yet provide enough information to be a basis for analysis, decision making, and hence risk reduction.[84]

This component approach to system design is integral in developing systems that incorporate data and functionality from external legacy systems. As stated in the quote above, software architectural considerations are not typically concerned with low-level component construction like algorithms or language selection. Instead, architectural design can be thought of as an assembly of puzzle pieces (functional components) with a set of rules that define how the pieces fit together and how separate puzzles are connected to create a larger construct.

Without adequate architectural planning that includes risk and requirements, software architecture can become a haphazard mesh of functionality that is poorly organized and prone to security and maintenance issues. Early HSIN development has been noted for lacking adequate architectural planning, which has resulted in HSIN's difficulty in connecting several important external data sources (RISSNet for example):

---

[83] IEEE Computer Society. Standards Coordinating Committee. et al., *IEEE Standard Glossary of Software Engineering Terminology* (New York, NY: Institute of Electrical and Electronics Engineers, 1990).

[84] Len Bass, Paul Clements, and Rick. Kazman, *Software Architecture in Practice*, SEI series in software engineering (Reading, Mass: Addison-Wesley, 1998), 21.

> [DHS] is missing, either in part or in total, all of the key elements expected to be found in a well-defined architecture, such as descriptions of business processes, information flows among these processes, and security rules associated with these information flows, to name just a few... Moreover, the key elements that are at least partially present in the initial version were not derived in a manner consistent with best practices for architecture development... As a result, DHS does not yet have the necessary architectural blueprint to effectively guide and constrain its ongoing business transformation efforts and the hundreds of millions of dollars that it is investing in supporting information technology assets.[85]

Solid architectural design planning helps developers manage complex systems by segmenting or partitioning functionality into manageable components. Best practices in software architectural design described in this section produces systems that are extensible and interoperable, built using data agnostic communication to maximize information sharing, and are easier to maintain and secure.

1. Evolving Need to Manage Complexity

Early government and industrial computer systems were primarily designed to facilitate existing business processes. Applications were developed to facilitate data input and retrieval for use within a single organization. These early systems had little or no ability to communicate with external organizations or between agencies.[86] By the late 1980s, increased power and availability of desktop computers and networked communications introduced the ability to share information and functionality to improve information sharing, reduce development cost and increase productivity. However, the complexity of

[85] Government Accountability Office. and Committee on Government Reform. Subcommittee on Technology, "Homeland Security Efforts Under Way to Develop Enterprise Architecture, But Much Work Remains: Report to the Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Committee on Government Reform, House of Representatives," 2.

[86] National Research Council (U.S.). Committee on the Fundamentals of Computer Science: Challenges and Opportunities, "Computer Science Reflections on the Field, Reflections from the Field," 2004, 1, http://www.netlibrary.com/urlapi.asp?action=summary&v=1&bookid=123466.

integrating legacy system functionality and data has proved tremendously challenging for both government and industry. Modern software architectural designs manage the complexity of large distributed systems by partitioning and layering functionality.

A core aspect of modern software architecture is the concept of component-based partitioning that reduces system complexity. To illustrate componentized complexity, system designers often explained the concept using a dice metaphor. Consider a single-sided die with each of the six sides representing a possible condition or state. Of course, a single six-sided die can have one of six states when thrown, one through six. Three six-sided die can have 216 possible states, which is 36 times more complex than a single die (216/6).[87]

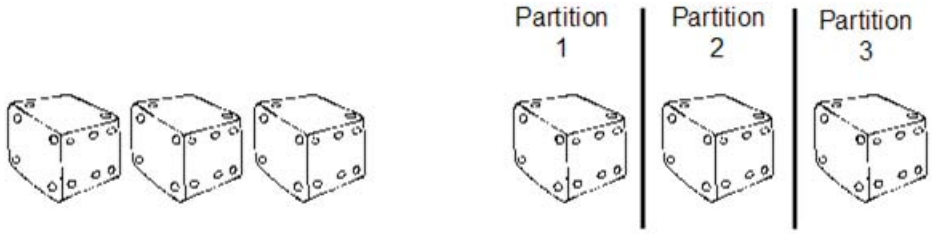

Figure 4. Dice Partitions Example[88]

To reduce the complexity of a multiple state system, each of the three die can be partitioned into a three component subsystems. Each partitioned die has only six possible states; with three partitions, the number of possible states is reduced to 18 (6+6+6). With all three dice functioning as a single system, "you would need to examine 216 different states, checking each for correctness." With each die examined as a separate system, you need only to examine six

[87] Roger Sessions, "A Better Path to Enterprise Architectures," *Microsoft Developers Network*, April 2006, http://msdn.microsoft.com/en-us/library/aa479371.aspx.

[88] Ibid.

different states to ensure the first partition is correct, another six states to ensure that the second partition is correct, and other six states to ensure that the third partition is correct.[89]

Table 3 extends the number of die to nine to show how rapidly complexity increases for non-partitioned systems compared to maintaining partitioned functionality. The goal of software component models is the same as the dice metaphor: separate complex system functionality into partitions or components with the goal of reducing system complexity with (as explained next) a side benefit of component reuse and extensibility. Partitioning also reduces the complexity of risk management, since each partition has a predetermined effect on other components. Risks can then be compartmented to support distributed risk models like SEI's MOSAIC, discussed earlier in this chapter, which is particularly well suited to managing risk associated with partitioned complexity.

S	D	S x D (partitioned)	S^D (non-partitioned)
6	1	6	6
6	2	12	36
6	3	18	126
6	4	24	1,296
6	5	30	7,776
6	6	36	46,656
6	7	40	279.936
6	8	46	1,679,616
6	9	52	10,077,696

Table 3. Partitioned and Non-Partitioned System States[90]

[89] Sessions, "A Better Path to Enterprise Architectures."

[90] Ibid.

2. Basics of Componentized Design Principles

Modern component-oriented architectures often subdivide application development into tiers that also help segment functionality. Segments typically include a presentation or user interface (UI) layer, a business logic layer and a data layer. Separating major functionality into layers or tiers comprised of components facilitates architectural concepts like interoperability, reusability and extensibility.

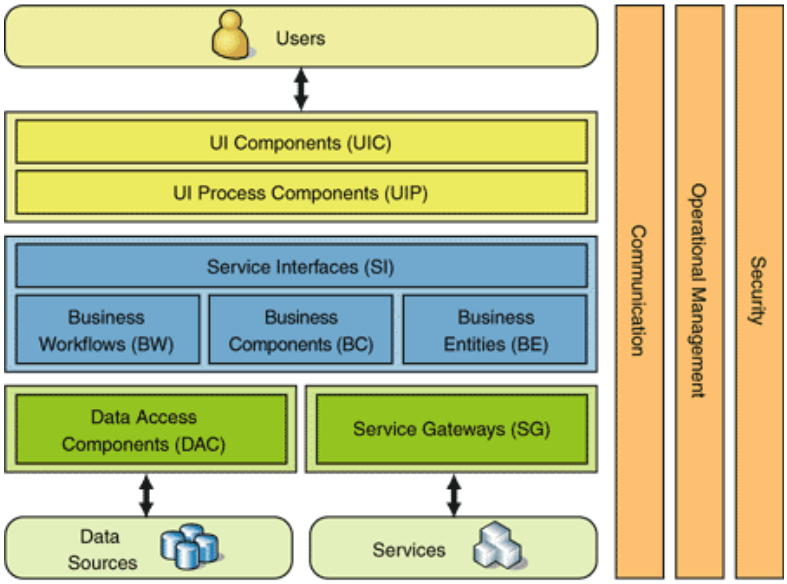

Figure 5. Tiered Application Layers[91]

The user interface layers do one thing and one thing only—interact with system users. As illustrated in Figure 4, user interface (UI) components and the associated process components do not contain business logic or data access code; they simply provide a means for users to enter data and display information. For an example, consider the desktop application in Figure 6. The UI provides a method for users to add, edit or remove data displayed in the table. The data table contains information that can be sorted by price, alphabetically, by

[91] "Three-Layered Services Application," *Microsoft Developers Network*, n.d., http://msdn.microsoft.com/en-us/library/ff648105.aspx.

item name or filtered to exclude information. The table and other interface items are separate components that can be reused in other applications since they do not contain any logic or data functionality. Instead, the displayed information is fed to the UI via the business logic layer.

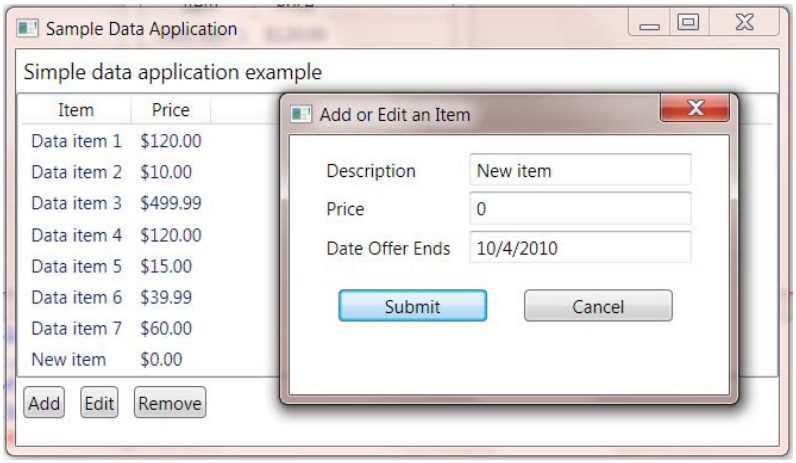

Figure 6. Sample Data Application[92]

The business logic layer contains methods that give meaning to raw data obtained from the database and feed to the UI. For example, if the user requests (through the UI) a list of customers in the Northwest, the business logic layer will query the data layer, manipulate the information depending on the business rules of the organization and feed the result to the UI. Since the business logic is comprised of separate components, business logic can be easily modified or replaced without having to significantly alter the UI or data access components.

The same is true for the data layer. If new methods for retrieving and manipulating data are developed in the future, additional functionality can be added to the data layer without impacting the business logic or UI. Modifying or replacing UI, business or data functionality is much more difficult in systems that do not implement a componentized, layered architecture.

[92] Microsoft Visual Studio v10 demonstration application.

a. Reusability

One advantage of using a layered and componentized system architecture is the ability to create generalized software components that are useable in other unrelated systems. For example, a business object designed to retrieve travel expense data, then calculate and return total cost may be reusable in other systems, even if they have different user interface and database system. Component use and reuse also has the potential to reduce development time and expense by giving developers the option to contract or purchase commercial components that can be plugged in to existing systems.

b. Extensibility

Another advantage of a modular architecture is the ability to add functionality as user needs change over time. Component-based systems contain communication methodologies that allow external components to be added as required to meet future needs. For example, the architecture design selected by developers may include an Application Programmer Interface (API) that creates source code communication pathways to otherwise non-compatible system functionality. At a higher level, extensibility may simply mean the ability to link to a separate organization's system to share functionality. An example of extensibility is HSIN Next Generation's video conferencing system, "Connect," which is a separate and complete Adobe product seamlessly integrated into the HSIN system.

c. Interoperability

Interoperability for software development is "the ability of two or more entities to communicate and cooperate regardless of differences in the implementation language, the execution environment, or model of abstraction."[93] The concept differs from extensibility in that interoperable architecture primarily

[93] M. Madiajagan and B. Vijayakumar, "Interoperability in Component Based Software Development" (2006): 69.

involves communication with external systems at the data layer. This is particularly useful for systems that are required to query external legacy systems. An organization wishing to share legacy system data may create a data access module that is able to communicate through an agnostic text-based communication such as XML. External applications can send and receive data via the interoperability incorporated into the system's architecture.

3. Architectural Frameworks

To solve the problem of data integration and to foster effective business practices, dozens of computer system and network-centric architectural frameworks have come in and out of favor over the past twenty years. The common goal of these frameworks is to separate or componentize complex business processes in order to ensure each piece of the puzzle is working efficiently and fits the needs of the enterprise. One of the most commonly used frameworks to reduce system complexity is Enterprise Architecture (EA) and its more recently developed cousin, Service Oriented Architecture (SOA), each with the goal of simplifying complex systems and fostering information and functionality sharing.

a. Enterprise Architecture

Enterprise Architecture (EA), also known as the Zachman Framework, was introduced to the software system development community in 1987 with the publishing of J. A. Zachman's article, "A Framework for Information Systems Architecture," in the IBM System Journal.[94] The EA concept was created to solve two problems: "System complexity—organizations were spending more and more money building IT systems; and poor business alignment—organizations were finding it more and more difficult to keep those

[94] John A. Zachman, *A Framework for Information Systems Architecture* (Los Angeles, Calif.: IBM Los Angeles Scientific Center, 1986).

increasingly expensive IT systems aligned with business needs."[95] Zachman describes his framework as "simply a logical structure for classifying and organizing the descriptive representations of an Enterprise that are significant to the management of the Enterprise, as well as to the development of the Enterprise's systems."[96] EA provides an intellectual framework for many of the system architectural designs used today. Zachman demonstrated that enterprise data, function, network, people, time and motivation are all viewed differently, based on the individual's business model, system model and technology model perspective. Using Zachman's framework, a system's architecture is considered functional only if meets the needs of each perspective in a way that melds business processes into a useful componentized structure.

Expanding from Zachman's original concepts, the Department of Defense, created the Technical Architecture Framework for Information Management (TAFIM) in 1991 and implemented in 1994.[97] The first published DoD TAFIM document identified services, standards, concepts and components that guide the development of architectural design patterns.

The success of TAFIM prompted Congress to pass the Clinger-Cohen Act of 1996, also known as the Information Technology Management Reform Act, "which mandated that all federal agencies take steps to improve the effectiveness of their IT investments."[98] Management of the program was eventually passed to the Office of Budget Management (OMB), where it was dubbed the government enterprise program the Federal Enterprise Architecture

---

[95] Roger Sessions, "A Comparison of the Top Four Enterprise-Architecture Methodologies," *Microsoft Developers Network*, n.d., http://msdn.microsoft.com/en-us/library/bb466232.aspx.

[96] John A. Zachman, *Enterprise Architecture and Legacy Systems: Getting Beyond the Legacy* (Zachman International, 1996), 1, http://www.ies.aust.com/papers/zachman1.htm.

[97] United States Dept. of Defense and United States. Defense Information Systems Agency, "Technical architecture framework for information management," 1996.

[98] Sessions, "A Comparison of the Top Four Enterprise-Architecture Methodologies."

(FEA). The federal government's goal for FEA is to divide process functionality into core business and global enterprise services that are available to each agency as needed.

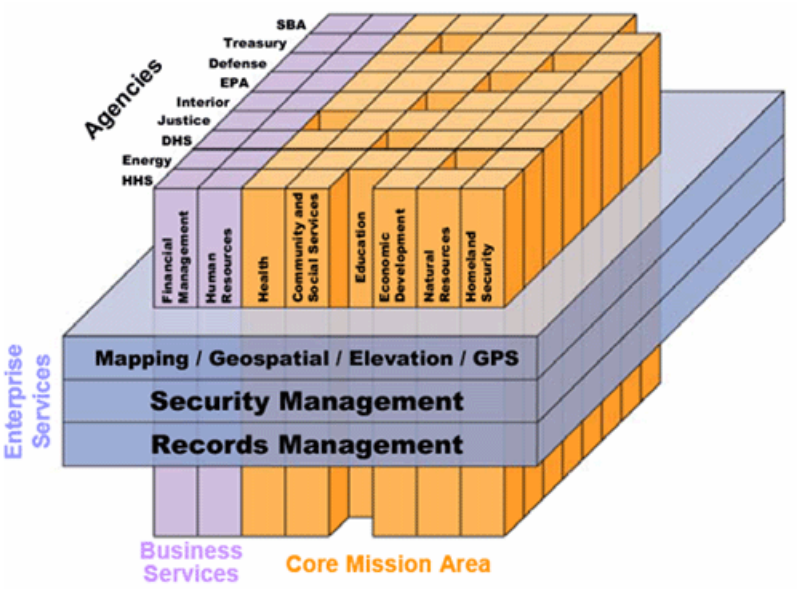

Figure 7. FEA Segment Map[99]

As Figure 6 shows, FEA core mission areas describe functionality used within an agency, such as Health or Education. Enterprise services, like security, records management and mapping, are common components used across the enterprise. This framework also ensures all government agencies have a common lexicon for describing these services. This helps facilitate component architecture communication and is described in the FEA Consolidated Reference Model Document Version 2.3.[100]

[99] Office of Management and Budget, *Federal Enterprise Architecture Practice Guidance* (White House, 2007), 3, http://www.whitehouse.gov/sites/default/files/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf.

[100] Sessions, "A Comparison of the Top Four Enterprise-Architecture Methodologies."

b. Service-Oriented Architecture

One of DHS's most difficult challenges for HSIN is to create an enterprise architecture that readily enables data and information sharing with legacy systems. Creating an effective system that bridges data across independent agencies has so far remained an elusive goal for DHS.

Until recently, sharing separate legacy system databases meant feeding data from the various systems into a central repository. The main challenge of feeding a central database is in developing intermediary database and code able to communicate with each independent data structure. To meet these challenges and leverage modern network communication technologies, Service Oriented Architecture, incorporating EA principles, helps simplify legacy system data and functionality sharing. The SEI explains it this way:

> The reality is that service-oriented architecture (SOA) is currently the best option available for systems integration and the leveraging of legacy systems. According to a 2007 Gartner Group report, 50% of new mission-critical operational applications and business processes were designed in 2007 around SOA, and that number will be more than 80% by 2010. While the technologies to implement SOA will probably change over time, one concept will remain: SOA promises a way to design systems that enables cost-efficiency, agility, adaptability, and the leveraging of legacy investments.[101]

From a high-level perspective, SOA is a componentized architecture that fosters the EA concept of incorporating business processes through modern communication and network protocols. SOA is closely tied to the tiered application model that simplifies reuse, extensibility, and interoperability by creating applications out of loosely coupled services designed to connect legacy systems. These services are typically delivered via web clients, but can also be delivered from service components to desktop application clients.

c. SOA Practical Example

SOA services provide functionality to any authenticated calling application or system designed to consume SOA services. Suppose Agency A maintains a 10+ -year-old database containing customer addresses designed to be accessed using desktop applications within the organization. Agency B maintains a similar database of customer data and uses this information to generate various reports. Without a service-oriented architecture, sharing information between agencies likely requires custom-built data access procedures for each agency with which it wishes to share information

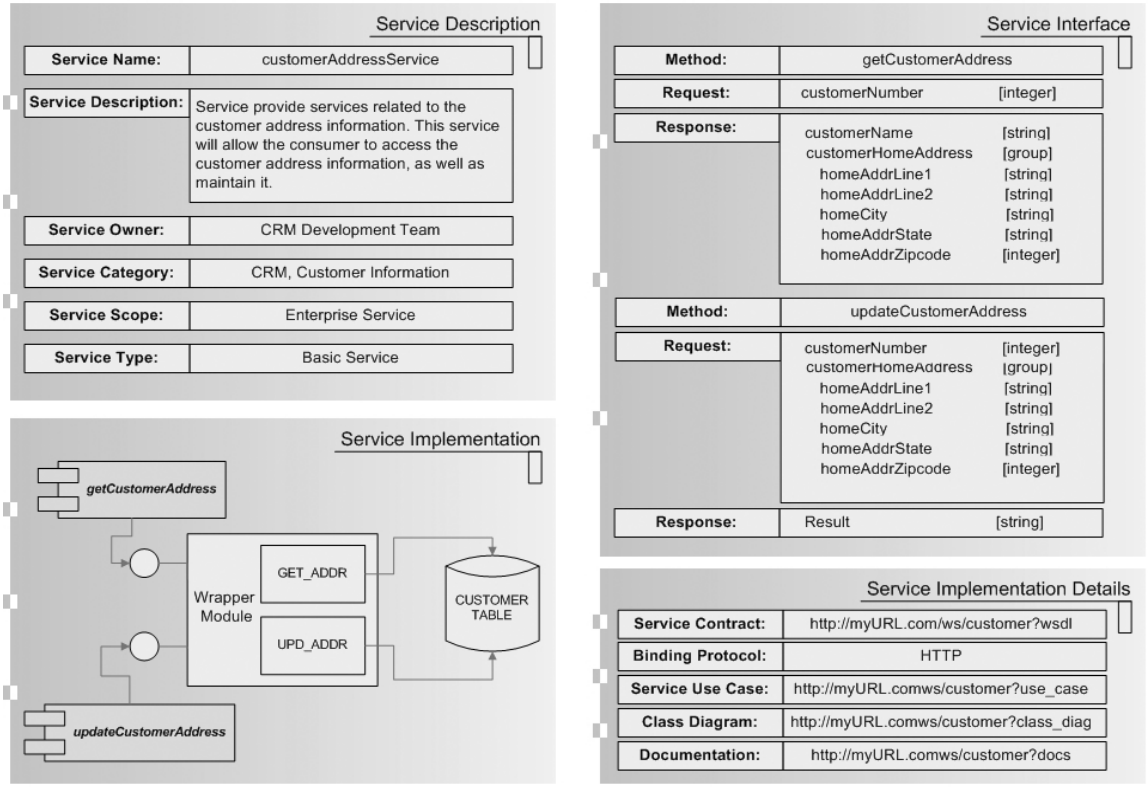

Figure 8. Sample SOA Service Design Specification[102]

[101] Software Engineering Institute, "Migrating Legacy Systems to SOA Environments – eLearning," n.d., http://www.sei.cmu.edu/training/v06.cfm.

[102] Software AG, "Your Guide to SOA Success," n.d., http://communities.softwareag.com/ecosystem/communities/public/businesscommunity/SOA/download_page_0002.html?overview=/public/businesscommunity/SOA/index.html&overview-page=0.

A SOA-based architecture allows Agency A and B to create services that hide the details necessary to access their proprietary database. As illustrated in Figure 8, Agency A could create a high-level service called CustomerAddressServices that contains a getCustomerAddress method. Agency B sends a service request via secure XML and receives the requested data via XML data structure.[103] Agency B does not need to know anything about the database implemented by Agency A other than how to make the request and the format to expect in the response (defined in the service specification). Complexity associated with database access is handled by the serving agency's service. If fifteen agencies need to share information, all fifteen can create and publish services to query their data, even if each agency uses radically different data architecture. As long a standard communication protocol is used (typically XML), retrieving information from one application to another is trivial.

d. Other SOA Considerations

Creating a system architecture strategy based on SOA requires an extensive initial planning approach when compared to traditional system designs. Exposing functionality via services takes additional planning and risk management in order to determine what data or functions to provide, how to limit access and provide secure data channels between client and service application.

For multi-agency SOA implementations, a plan must exist to ensure SOA methodologies and practices are standardized to reduce complexity across the enterprise. Seemingly simple decisions like method-naming conventions and documentation standards can become confused if each agency creates separate policies. The U.S. government sets IT architectural standards through the OMB and the Office of E-Government that are communicated through Federal Enterprise Architecture (FEA) documentation. FEA incorporates SOA into their reference model construct and establishes a baseline for data, performance, business and service component development. For example, the FEA data

[103] Software AG, "Your Guide to SOA Success."

reference model specifications outlines, “XML schema, the data context defined by XML namespaces, and data sharing expressed via XML-based request exchange patterns used within a the Web service (SOA) framework.”[104] U.S. government agencies wishing to share data via services must comply with this FEA model.

C. CONCLUSION

The previous chapter established the need for a dedicated project management team for the development or acquisition of large, complex systems. The project team needs time and resources to understand and document the problem the system is trying to solve. Gathering and managing user requirements is a good first step toward developing a useful system.

As this chapter shows, acquisition managers also need the time to adequately plan and manage risk, and establish a component-based architecture that reduces integrated-system complexity. Unfortunately, these areas are sometimes given minimal resources when time is critical. However, skipping these steps can actually create delays if contingency plans and component structures are not in place early in a project’s life cycle. Integrating expandability, reusability and interoperability through architectures like SOA can help ensure newly developed systems remain viable as technology changes over time. Good, up-front enterprise architecture planning that manages complexity can help achieve this goal.

[104] Office of Management and Budget, *Federal Enterprise Architecture The Data Reference Model Version 2.0* (White House, 2005), http://www.whitehouse.gov/sites/default/files/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf.

VI. FINAL ANALYSIS

It is impossible to design a system so perfect that no one needs to be good.

—T.S. Eliot

Designing and acquiring a government software system is certainly a challenging task. If government applications were only required to function on a single system for a single user, the complexity associated with networks and data sharing would not be an issue. Of course, government-acquired software and data do not live in a self-contained box. Today's software must be designed to accommodate multiple, geographically separated role-based users, utilize local and distributed networks, and provide a means to share and use data and functionality. In many cases, sharing must be accomplished between systems that were not originally designed to accommodate outside data requests. If government systems do not meet these information-sharing challenges, as well as provide a responsive and reliable interface, newly acquired systems can and will fail. These failures cost taxpayers millions of dollars, and must be avoided.

While HSIN is not a failed system, it has struggled to meet its mandate to provide users with a solid collaboration and data-sharing platform. Recent upgrades and integration of COTS components have made it more useful. Also, plans for future deployment of Next Generation components are promising. However, studying how and why HSIN initially failed to meet user's expectations provides valuable insight into how and why systems falter and sometimes fail. Additionally, lessons learned from both failed and successful software acquisition programs can help form best practices that bring 20/20 hindsight forward to future project success.

A. UNDERSTANDING THE PAST TO PROMOTE FUTURE ACQUISITION SUCCESS

HSIN is an example of the need to create a new system that integrates several complex legacy systems not originally designed to share information. As initial HSIN development illustrates, the rush to deliver a complex system can result in inadequate planning in key areas needed for project success. Studying HSIN demonstrated that program managers should be assigned and adequately staffed to ensure that requirements, risk and architectural planning occur throughout the project's life cycle. This thesis concludes with a summary of these three areas and demonstrates that these important planning processes can be the difference between a successful or failed project.

1. Requirements are Central to Software Planning

SEI research shows that as many as 20 percent of large, multi-million-dollar projects are never completed or, when delivered, do not meet its user's needs.[105] Many of these projects fail due to a lack of requirements planning that make establishing minimum initial functionality and milestones difficult or impossible. Without a comprehensive requirements management plan, projects easily lose focus on the problem the system was originally attempting to solve.

Developing a strategy to effectively deliver functionality that meets the user's current and future needs requires an understanding of requirements processes that have worked in past projects. The IEEE Software Journal and SEI provide outstanding best practices that ensure requirements plans match user needs with a functionality set that supports milestone planning and easily feeds risk and architecture development plans.

Common in best practice requirements methodologies (CMMI and IEEE) include user requirements elicitation through user questionnaires, discussion groups, operational scenarios and/or a review of business process documents.

[105] Software Engineering Institute, Carnegie Mellon: Software Development.

This phase of planning also consists of a survey of existing technology, currently in use, that may match stakeholder functionality requirements. Results of user elicitation can then be prioritized and analyzed to develop an initial functionality document that is validated between the stakeholder and developers. Functionality that is considered essential becomes part of the initial operation capability plan, and the remaining items are set aside until future iterations are planned. From here, risk management and architectural design planning can then further the planning process.

2. Risk Management for Integrated Systems

It is difficult to imagine a complex software acquisition that does not experience unexpected problems during development and deployment. Taking the time to plan for potential stumbling blocks can help minimize a potential problem's effect on the project. Not planning for risk can appear to save planning time but often results in project teams fighting fires during development that lead to delays and unintended consequence. The increasing complexity of integrated software systems make planning for and mitigating risk a critical part of the planning process.

Good risk management practices require planners to identify potential problems that could derail the project by determining the probability and expected loss should the risk event occur. Dr. Barry Boehm's risk equation helps planners quantify identified risks in order to develop a prioritized list from most likely and costly risks to low probability and less costly risks. From here, individual risks should be analyzed to determine their effect on other risks that can negatively impact the project. Once initial risk planning is complete, the system should be reevaluated for risks as project development progresses and contingency plans are executed (as required).

SEI's MOSAIC expands the basic risk-planning approach to methods better suited to managing complex distributed systems. This approach helps planners evaluate risk and uncertainty across multiple systems with multiple

variables by analyzing organizational processes. MOSAIC's helps manage complex risk problems by segmenting processes, determining process risk and the probably of impact on entire enterprise. This risk approach is particularly useful for programs like HSIN, integrating several separate systems that each have their own set of potentially interrelated risks.

3. Architecture to Manage Complexity

Modern software systems are becoming increasingly complex. Integrating user requirements and risk mitigation requires an architectural plan that effectively manages complexity. Systems that are pieced together over time by different developers tend to be difficult to maintain, prone to security issues, and do not communicate well with other systems. An initial investment in best practice architectural planning can help ensure systems can easily share information and functionality and expand as requirements change.

Componentized architecture like SOA is becoming an industry standard and is fully incorporated into the U.S. government's Federal Enterprise Architecture framework. FEA, along with solid risk and requirements planning, have the potential to help avoid problems encountered by HSIN and to increase the probability of future government software acquisition success.

APPENDIX

A. HOMELAND SECURITY INFORMATION NETWORK AND THE DEEPWATER HORIZON GULF OIL SPILL

United States infrastructure resilience depends on an effective government and private industry response when a disaster strikes. How efficiently government and industry responds to a national crisis plays a tremendous role in the degree and length of impact on the nation. The time it takes to initiate recovery operations often depends on the government's ability to coordinate actions with the industry that owns the recovery infrastructure.

Ensuring this coordination and communication takes place during a disaster is primarily the responsibility of the Department of Homeland Security (DHS). Given the risks associated with extraction and transportation of oil and gas, the U.S. energy sector requires well-coordinated and planned emergency actions to reestablish flow when disruptions occur. The April 2010 explosion of the oil platform Deepwater Horizon in the Gulf of Mexico provides an example of how a disaster in this sector can affect the economy, commodity levels and the environment.

HSIN in its current form allows emergency responders to coordinate recovery plans, delegate responsibility, and follow up on actions that support a rapid disaster response. During the early stages of a national disaster, determining the command structure is key to an effective response. DHS is responsible for establishing the crisis-response command structure and uses HSIN to coordinate actions that begin infrastructure and recovery operations.

On 20 April 2010, a Beyond Petroleum (BP)-owned oilrig platform, the Deepwater Horizon, exploded in the Gulf of Mexico, killing eleven oil workers and creating one of the worst oil spills on record in the U.S. The explosion resulted in as much as 62,000 barrels of oil gushing into the gulf per day, creating the potential for irreparable damage to several critical Gulf Coast natural resources.

At the time of the disaster, damage to the region's fishing and tourism industries was estimated to cost the local economy several billion dollars.[106]

Immediately after the Deepwater Horizon explosion, DHS dispatched the U.S. Coast Guard to rescue 126 platform workers and established a command center for the, "16 federal departments and agencies responsible for coordinating emergency preparedness and response to oil and hazardous substance pollution events." [107] The Department of Interior's (DOI) role in the emergency response was to ensure BP could provide an adequate response plan to stop the leak and communicate progress to DHS and to the President of the United States. For daily operations, the Coast Guard was assigned the mission of coordinating Regional Response Teams and to act as on-scene incident command to orchestrate the actions of the Defense Department, Environment Protection Agency, National Oceanic and Atmospheric Administration, Small Business Administration, Department of Labor and the National Parks Service.

To facilitate information flow between these agencies, DHS established an HSIN portal called MC252 under the Emergency Management Community of Interest. MC252 allowed the Department of the Interior and Coast Guard to establish a secure collaboration environment for all involved government entities. The portal also highlights some of the enhancements DHS put in place during HSIN's "Next Generation" upgrade, which was still in progress at the time of the disaster.

One of these Next Generation components is the recently revamped common operating picture (COP) system. This component provided users with real-time, constantly updated information concerning the spill. Another purpose of the Deepwater Horizon Incident COP is to provide a tracking system for deployed forces responding to the incident. Service members from all branches

---

[106] John Kennedy, "Economist: Oil disaster could cost Florida economy 39,000 jobs, $2.2 billion," *Palm Beach Post*, June 7, 2010, http://www.palmbeachpost.com/news/state/economist-oil-disaster-could-cost-florida-economy-39-732979.html.

[107] "Deepwater BP Oil Spill," *Whitehouse.gov*, n.d., http://www.whitehouse.gov/deepwater-bp-oil-spill.

of the military services, including the Coast Guard and National Guard mobilized to respond to the crisis. The COP provided a way for each service to locate and communicate with other responders.

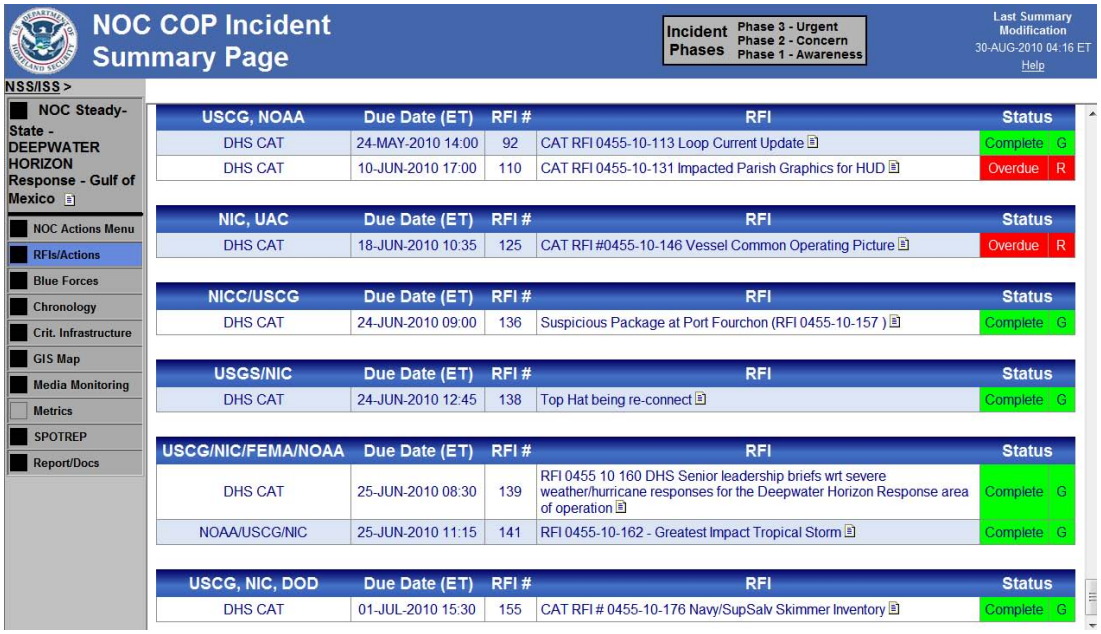

Figure 9. HSIN Common Operating Picture for MC252[108]

As Figure 8 shows, the MC252 portal also tracks media reports and internal requests for information. Responsibility is assigned for each task, along with updated status information. Each assigned task has a tracking date, source, and resolution information once complete. Users can search questions already asked before submitting additional requests. The MC252 portal also includes maps, critical infrastructure information, media monitoring and a Request for Information (RFI) tracker. A library of reports and other documents, along with a MC252 specific video meeting application (Adobe Connect), was added to this event specific portal.

108 "Common Operating Picture," *Homeland Security Information Network*, n.d., https://government.hsin.gov/default.aspx.

B. CONCLUSION

Well-coordinated interagency action following a disaster can shorten the time required to begin recovery operations and bring systems back online. Despite its rocky start in 2004, HSIN upgrades seem to have increased its usefulness as an interagency collaboration tool. The use of HSIN to create the MC252 portal and its use among government agencies for disaster recovery indicate DHS's mandate to improve government communication is beginning to occur.

LIST OF REFERENCES

Alberts, Christopher J., and Audrey J. Dorofee. “Risk Management Framework.” Software Engineering Institute, Carnegie Mellon, August 2010. http://www.sei.cmu.edu/reports/10tr017.pdf (accessed October 29, 2010).

Andrus, D. Calvin. “The Wiki and the Blog: Toward a Complex Adaptive Intelligence Community,” *Studies in Intelligence* (2005). http://ssrn.com/abstract=755904 (accessed October 29, 2010).

Arizona State University. “Question List for Software Risk Identification in the Classroom” (n.d.) http://www.eas.asu.edu/~riskmgmt/qlist.html (accessed October 29, 2010).

Bass, Len, Paul Clements, and Rick. Kazman. *Software Architecture in Practice. SEI: Series in Software Engineering.* Addison-Wesley, 1998.

Boehm, Barry W., and Ez. Nahouraii. IEEE Computer Society, “Software Risk Management: Principles and Practices.” IEEE Computer Society Press, 1989: 426–435. http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/RiskManagement/PrinciplesandPractices.pdf (accessed October 29, 2010).

CMMI for Acquisition, Version 1.2. Carnegie Mellon University, Software Engineering Institute, 2007. http://www.sei.cmu.edu/reports/07tr017.pdf (accessed October 29, 2010).

Common Operating Picture. Homeland Security Information Network (n.d.) https://government.hsin.gov/default.aspx (accessed October 29, 2010).

Data and Analysis Center for Software. https://www.thedacs.com/ (accessed October 29, 2010).

———. “Requirements Engineering,” n.d. https://www.thedacs.com/databases/url/key/5086 (accessed October 29, 2010).

———.“Risk Management,” 2010. https://www.thedacs.com/databases/url/key/270 (accessed October 29, 2010).

———. Software Acquisition Gold Practice, 2010. https://goldpractice.thedacs.com/practices/api/ (accessed October 29, 2010).

Deepwater BP Oil Spill." *Whitehouse.gov*, (n.d.) http://www.whitehouse.gov/deepwater-bp-oil-spill (accessed October 29, 2010).

Department of Homeland Security. "About Homeland Security Information Network" (n.d.) http://www.dhs.gov/files/programs/gc_1156888108137.shtm (accessed October 29, 2010).

Dorofee, Audrey, and Christopher Alberts. "Rethinking Risk Management: NDIA Systems Engineering Conference." Carnegie Mellon University, Software Engineering Institute, 2009. http://www.sei.cmu.edu/library/abstracts/risk/upload/dorofeetutorialworkbookndia09_8819-1.pdf (accessed October 29, 2010).

Enterprise Architecture Framework Version 2.0. Information Sharing Environment, 2008. http://www.ise.gov/docs/eaf/ISE-EAF_v2.0_20081021.pdf (accessed October 29, 2010).

Final Report: Homeland Security Information Network Advisory Committee Meeting, February 10–12, 2009. The Department of Homeland Security (March 27, 2009). http://www.dhs.gov/xlibrary/assets/hsinac_mtg_2009-2-1012.pdf (accessed October 29, 2010).

Government Accountability Office. "Homeland Security Opportunities Exist to Enhance Collaboration at 24/7 Operations Centers Staffed by Multiple DHS Agencies: Report to Congressional Requesters," 2006. http://purl.access.gpo.gov/GPO/LPS76414 (accessed October 29, 2010).

———. "Information Technology Management Improvements Needed on the Department of Homeland Security's Next Generation Information Sharing System: Report to Congressional Requesters," 2008. http://purl.access.gpo.gov/GPO/LPS104962 (accessed October 29, 2010).

———. "Information Technology Numerous Federal Networks Used to Support Homeland Security Need to be Better Coordinated with Key State and Local Information-Sharing Initiatives: Report to the Chairman, Committee on Homeland Security, House of Representatives," 2007. http://purl.access.gpo.gov/GPO/LPS82926 (accessed October 29, 2010).

———. “Information Technology Homeland Security Information Network Needs to be Better Coordinated with Key State and Local Initiatives: Testimony Before the Subcommittee on Intelligence, Information Sharing and Terrorism Risk Assessment, Committee on Homeland Security, House of Representatives,” 2007. http://purl.access.gpo.gov/GPO/LPS83332 (accessed October 29, 2010).

———. “Homeland Security Efforts Under Way to Develop Enterprise Architecture, But Much Work Remains: Report to the Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Committee on Government Reform, House of Representatives,” 2004. http://purl.access.gpo.gov/GPO/LPS53957 (accessed October 29, 2010).

Homeland Security Information Sharing Act of 2002. Public Law 107–296, November 25, 2002.

Hubert F. Hofmann. “Requirements Engineering as a Success Factor in Software Projects.” *IEEE Software*, July 1, 2001.

IEEE Computer Society. Software Engineering Standards Committee, and Institute of Electrical and Electronics Engineers. *IEEE Recommended Practice for Software Acquisition*. New York, NY: Institute of Electrical and Electronics Engineers, Inc., 1994.

———. Standards Coordinating Committee, Institute of Electrical and Electronics Engineers, IEEE Standards Board, and American National Standards Institute. *IEEE Standard Glossary of Software Engineering Terminology*. New York, NY: Institute of Electrical and Electronics Engineers, 1990.

John and Mary R. Markle Foundation. *Mobilizing Information to Prevent Terrorism: Accelerating Development of a Trusted Information Sharing Environment.* New York City: The Markle Foundation, 2006.

Kennedy, John. “Economist: Oil disaster could cost Florida economy 39,000 jobs, $2.2 billion.” *Palm Beach Post*, June 7, 2010. http://www.palmbeachpost.com/news/state/economist-oil-disaster-could-cost-florida-economy-39-732979.html (accessed October 29, 2010).

Lessons Learned from a Large, Multi-Segment, Software-Intensive System. *Software Engineering Institute, Carnegie Mellon*, n.d. http://www.sei.cmu.edu/library/abstracts/reports/09tn013.cfm (accessed October 29, 2010).

Linaje, Marino, Juan Carlos Preciado, and Fernando Sanchez-Figueroa. "Engineering the Web Track - Engineering Rich Internet Application User Interfaces over Legacy Web Models." *IEEE internet computing.* 11, no. 6 (2007): 53.

Madiajagan, M., and B. Vijayakumar, "Interoperability in Component Based Software Development" *World Academy of Science, Engineering and Technology,* October 2006, Issue 22. 68–76.

Markle Foundation Task Force. "Nation at risk policy makers need better information to protect the country," 2009. http://www.markle.org/events/20090310_nar/20090304_mtf_report.pdf (accessed October 29, 2010).

Meeting Minutes: Homeland Security Information Network Advisory Committee Inaugural Meeting, October 30 – November 1 2007. *The Department of Homeland Security*, December 28, 2007. http://www.dhs.gov/xlibrary/assets/hsinac_inauguralmtg_2007-1030-1101.pdf (accessed October 29, 2010).

National Research Council (U.S.). Committee on the Fundamentals of Computer Science: Challenges and Opportunities. "Computer Science Reflections on the Field, Reflections from the Field," 2004. http://www.netlibrary.com/urlapi.asp?action=summary&v=1&bookid=123466 (accessed October 29, 2010).

Office of Management and Budget. *Federal Enterprise Architecture Practice Guidance*. White House, The, 2007. http://www.whitehouse.gov/sites/default/files/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf (accessed October 29, 2010).

Office of Management and Budget. *Federal Enterprise Architecture The Data Reference Model Version 2.0*. The White House, 2005. http://www.whitehouse.gov/sites/default/files/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf (accessed October 29, 2010).

Relyea, Harold, and Jeffrey W. Seifert. Congressional Research Service. *Information Sharing for Homeland Security a Brief Overview*. Washington, DC: Congressional Information Service, Library of Congress, 2005.

Sessions, Roger. "A Better Path to Enterprise Architectures." *Microsoft Developers Network* (April 2006). http://msdn.microsoft.com/en-us/library/aa479371.aspx (accessed October 29, 2010).

Sessions, Roger. “A Comparison of the Top Four Enterprise-Architecture Methodologies.” *Microsoft Developers Network* (May 2007) http://msdn.microsoft.com/en-us/library/bb466232.aspx. (accessed October 29, 2010).

Software AG. “Your Guide to SOA Success” (October 2010) http://communities.softwareag.com/ecosystem/communities/public/businesscommunity/SOA/download_page_0002.html?overview=/public/businesscommunity/SOA/index.html&overview-page=0 (accessed October 29, 2010).

Software Engineering Institute, Carnegie Mellon: Software Development, 2010. http://www.sei.cmu.edu/solutions/softwaredev/ (accessed October 29, 2010).

———. (n.d.), http://www.sei.cmu.edu/ (accessed October 29, 2010).

———. Software Development. “Migrating Legacy Systems to SOA Environments – eLearning,” (n.d.) http://www.sei.cmu.edu/training/v06.cfm (accessed October 29, 2010).

Software Program Manager’s Network. “The Little Book of Bad Excuses,” 1998. http://www.spmn.com/products_guidebooks.html (accessed October 29, 2010).

Software Technology Support Center. “Understanding Risk Management.” *CrossTalk* (February 2005). http://www.stsc.hill.af.mil/crosstalk/2005/02/0502stsc.html (accessed October 29, 2010).

Three-Layered Services Application. *Microsoft Developers Network*, (n.d.), http://msdn.microsoft.com/en-us/library/ff648105.aspx (accessed October 29, 2010).

U.S. Executive Office of the President. “National Strategy for Information Sharing: Successes and Challenges in Improving Terrorism-Related Information Sharing,” October 2007. http://handle.dtic.mil/100.2/ADA473664 (accessed October 29, 2010).

United States Dept. of Defense. Defense Information Systems Agency. “Technical Architecture Framework for Information Management,” 1996.

Westfall, Linda. “Software Risk Management.” *The Westfall Team*, 2001. http://www.westfallteam.com/Papers/risk_management_paper.pdf (accessed October 29, 2010).

Yim, Randall A. United States General Accounting Office. “National Preparedness Integrating New and Existing Technology and Information Sharing Into an Effective Homeland Security Strategy,” 2002. http://purl.access.gpo.gov/GPO/LPS34938 (accessed October 29, 2010).

Zachman, John A. *A Framework for Information Systems Architecture*. Los Angeles, Calif.: IBM Los Angeles Scientific Center, 1986.

Zachman, John A. *Enterprise Architecture and Legacy Systems: Getting Beyond the Legacy*. Zachman International, 1996. http://www.ies.aust.com/papers/zachman1.htm.

Zegart, Amy B. *Spying Blind: The CIA, the FBI, and the Origins of 911*. Princeton, N.J.: Princeton University Press, 2007.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California